

Michael M. Richter
Rosina O. Weber

Case-Based Reasoning

A Textbook

 Springer

Case-Based Reasoning

Michael M. Richter • Rosina O. Weber

Case-Based Reasoning

A Textbook

 Springer

Michael M. Richter
Fachbereich Informatik (retired)
Universität Kaiserslautern
Kaiserslautern, Germany

Rosina O. Weber
The College of Computing & Informatics
Drexel University
Philadelphia, PA, USA

ISBN 978-3-642-40166-4

ISBN 978-3-642-40167-1 (eBook)

DOI 10.1007/978-3-642-40167-1

Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013953284

© Springer-Verlag Berlin Heidelberg 2013

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

A statement of George Bernhard Shaw is essentially this:

“The wisdom of a person is not measured by the number of experiences but rather by the ability to make use of them”.

This characterises case-based reasoning (CBR) to a large degree. It is easy to record billions of experiences in a database. However, the central question is: If I have a problem, which experience should I consult and in what way? Humans have cultivated this in very clever ways. So, what prevents us from collecting experiences on a computer? Nothing: Collection is easily done. The problem is now to realise the second part of Shaw’s statement on a computer. This has been neglected for a long time and is still not a standard in university education.

This difficulty is obvious: The new problem is not exactly like that in the experience base, and even if we find a good one quite similar, it cannot be used exactly in the same way as it was used in the past.

This book presents a systematic approach. It has two goals:

- (1) to present rigorous and formally valid structures for precise case-based reasoning;
- (2) to introduce you to the universe of CBR applications through the understanding of techniques, methods, and tools for many quite different tasks.

The basic formalisms are:

- A process model. This describes on an abstract level the steps that one has to perform when working with CBR. This incorporates, in particular, what to do in order to find useful experiences and how to apply them once they are selected.
- The knowledge containers. They describe how and where knowledge is stored that is needed for executing the steps of the process model.

There are two ways of working with CBR. The experience-based view compares a new problem to a previous experience. The extended view does not use experiences; it searches for a solution of a problem by directly associating a solution to the problem. In both approaches all the techniques with respect to the process model and the knowledge containers are valid.

Chapter 1 introduces you to the organization of the book. The book includes core methods, advanced elements, a part dedicated to complex knowledge sources, and additions.

The book also presents many applications. Mostly, they are not given in full detail. However, they are used to guide you on how to proceed. Most of these examples result from actual projects and student work over a decade with the participation of the authors.

Despite the wealth of applications and the scientific research progress, CBR has played only a minor role in education. It is not among the general topics in the curricula of computer-related fields. One reason for this seems to be the lack of textbooks. In English, there is Kolodner's *Case-Based Reasoning* from 1993. This book is a thorough compendium of the first years of research in CBR.

There is also *Raciocínio Baseado em Casos*, in Portuguese, by Christiane Gresse von Wangenheim and Aldo von Wangenheim, published in 2003 (Curitiba: Editora Manole). This covers the process model and applications.

This textbook aims at changing this status quo. It is not only aimed at students but also at potential lecturers. The latter are supported in different ways, for instance, by structured text, exercises and tools. This book is also intended for professionals designing and developing CBR systems.

Acknowledgements

The authors are indebted to all of their collaborators and students. Much of the research relied on their work.

In the beginning there was much contribution to the structure and the content of the book by Klaus-Dieter Althoff, Ralph Bergmann, Thomas Roth-Berghofer, and Armin Stahl. They were involved in the first phase of the book project. In addition, the book benefited from their lectures. Special thanks go also to Sebastian von Mammen for contributions to Chap. 18 (Images); and to Sheuli Paul for contributions to Chap. 19 (Sensor Data and Speech). Ralph Traphöner supported us with CBRWorks.

During the years while this book was in preparation, Rosina Weber was partially funded by the US EPA Science to Achieve Results (STAR) Program and the US Department of Homeland Security University Programs, Grant #R83236201. In the 2010–2011 academic year, Rosina was on sabbatical leave, granted by her institution, the iSchool at Drexel, College of Information Science and Technology, Drexel University.

The authors would like to thank the work of the copyeditors, which was crucial in getting the text into its final shape.

Kaiserslautern, Germany
Philadelphia, USA

Michael M. Richter
Rosina O. Weber

Contents

Part I Basics

1	Introduction	3
1.1	About This Chapter	3
1.2	General Aspects	3
1.2.1	A Bit of the History	4
1.3	Intended Readers	6
1.4	Outline of the Book	7
1.4.1	Part I: Basics	7
1.4.2	Part II: Core Methods	9
1.4.3	Part III: Advanced Elements	11
1.4.4	Part IV: Complex Knowledge Sources	12
1.4.5	Part V: Additions	13
1.4.6	Chapter Structure	14
	References	15
2	Basic CBR Elements	17
2.1	About This Chapter	17
2.2	General Aspects	17
2.3	Case-Based Reasoning	17
2.4	Experiences and Cases	18
2.4.1	Parts of a Case	20
2.4.2	Problems	21
2.4.3	Solution Types	22
2.5	Case Representations	23
2.5.1	How Cases Are Represented	23
2.6	Case Bases	24
2.6.1	How Are Cases Organised?	25
2.7	Similarity and Retrieval	26
2.8	Reuse and Adaptation	30
2.9	Models of CBR	31
2.9.1	CBR Process Model	32

- 2.9.2 CBR Knowledge Model 34
- 2.10 Tools 37
- 2.11 Chapter Summary 38
- 2.12 Background Information 38
- 2.13 Exercises 39
- References 40
- 3 Extended View 41**
 - 3.1 About This Chapter 41
 - 3.2 General Aspects 41
 - 3.2.1 E-Commerce 42
 - 3.2.2 Recommender Systems 46
 - 3.3 Extended Model 47
 - 3.4 More Generalizations 49
 - 3.5 Tools 50
 - 3.6 Chapter Summary 51
 - 3.7 Background Information 51
 - 3.8 Exercises 51
 - References 52
- 4 Application Examples 53**
 - 4.1 About This Chapter 53
 - 4.2 General Aspects 53
 - 4.3 Analytic Tasks 55
 - 4.3.1 Classification 56
 - 4.3.2 Diagnosis 57
 - 4.3.3 Prediction 62
 - 4.4 Synthetic Tasks 64
 - 4.4.1 Configuration 64
 - 4.4.2 Planning 66
 - 4.4.3 Design 69
 - 4.5 Organisation-Oriented Applications 70
 - 4.5.1 Call Centres 70
 - 4.5.2 E-Commerce 72
 - 4.5.3 Knowledge Management 75
 - 4.5.4 Law 78
 - 4.6 Complex Knowledge Sources 80
 - 4.6.1 Texts 80
 - 4.6.2 Images 81
 - 4.7 Chapter Summary 82
 - 4.8 Background Information 82
 - 4.9 Exercises 83
 - References 84

Part II Core Methods

- 5 Case Representations 87**
 - 5.1 About This Chapter 87
 - 5.2 General Aspects 87
 - 5.2.1 Representation Layers 87
 - 5.2.2 Completeness and Efficiency 92
 - 5.2.3 Flat Attribute-Value Representation 93
 - 5.2.4 Complex Representations in General 95
 - 5.3 Tools 109
 - 5.4 Chapter Summary 110
 - 5.5 Background Information 110
 - 5.6 Exercises 110
 - References 111

- 6 Basic Similarity Topics 113**
 - 6.1 About This Chapter 113
 - 6.2 General Aspects 113
 - 6.3 Similarity and Case Representations 114
 - 6.3.1 Mathematical Models of Similarity 115
 - 6.3.2 Meaning of Similarity 115
 - 6.4 Types of Similarity Measures 125
 - 6.4.1 Counting Similarities 126
 - 6.4.2 Metric Similarities 129
 - 6.4.3 Comparisons 130
 - 6.4.4 Structured Similarities and Symbolic Arguments 130
 - 6.4.5 Transformational Similarities 133
 - 6.5 The Local-Global Principle for Similarity Measures 133
 - 6.5.1 Weighted Measures 134
 - 6.5.2 Local Measures 135
 - 6.5.3 Global Aspects 138
 - 6.5.4 Weights 138
 - 6.6 Virtual Attributes 141
 - 6.7 Which Similarity Measure Should I Use? 142
 - 6.8 Tools 144
 - 6.9 Chapter Summary 145
 - 6.10 Background Information 145
 - 6.11 Exercises 146
 - References 147

- 7 Complex Similarity Topics 149**
 - 7.1 About This Chapter 149
 - 7.2 Graph Representations and Graph Similarities 149
 - 7.2.1 Graph Isomorphism 149
 - 7.2.2 Subgraph Isomorphism 150
 - 7.3 Largest Common Subgraphs 150

7.3.1	Edit Operations	151
7.4	Taxonomic Similarities	153
7.5	Similarities for Object-Oriented Representations	156
7.6	Many-Valued Attributes	158
7.7	Similarity for Processes and Workflows	159
7.7.1	Similarities for Time Series	161
7.8	Tools	162
7.9	Chapter Summary	162
7.10	Background Information	163
7.11	Exercises	163
	References	164
8	Retrieval	167
8.1	About This Chapter	167
8.2	General Aspects	167
8.3	The Retrieval Task	168
8.3.1	Retrieval Errors	169
8.4	Basic Retrieval Methods	170
8.4.1	Query Generation	170
8.4.2	Filtering	170
8.4.3	Sequential Retrieval	171
8.4.4	Two-Level Retrieval	172
8.4.5	Geometric Methods	173
8.4.6	Voronoi Diagrams and k-Nearest Neighbours	173
8.4.7	Geometric Approximation	175
8.4.8	Geometric Filtering	176
8.4.9	Index-Based Retrieval	177
8.4.10	kd-Trees	178
8.4.11	Integration with Decision Trees	183
8.5	Tools	185
8.6	Chapter Summary	185
8.7	Background Information	185
8.8	Exercises	186
	References	186
9	Adaptation	189
9.1	About This Chapter	189
9.2	General Aspects	189
9.3	Rules	191
9.3.1	Preconditions	192
9.3.2	Actions	192
9.3.3	Types of Rules	193
9.3.4	Integrating Completion and Adaptation Rules	197
9.4	Adaptation Types	198
9.5	The Adaptation Process	199
9.5.1	Adaptation Sequences	200

- 9.5.2 Adaptation Planning 202
- 9.5.3 Learning Heuristics 209
- 9.5.4 Adaptation in More Complex Situations 210
- 9.6 Adaptation Using Several Cases 210
 - 9.6.1 Simple Numerical Adaptations 210
- 9.7 Adaptations Using the Solution Process 213
- 9.8 Quality Issues 216
- 9.9 Knowledge in the Adaptation Container 216
- 9.10 When Should Adaptation Be Considered? 217
- 9.11 Tools 217
- 9.12 Chapter Summary 217
- 9.13 Background Information 218
- 9.14 Exercises 218
- References 219
- 10 Evaluation, Revision, and Learning 221**
 - 10.1 About This Chapter 221
 - 10.2 General Aspects 221
 - 10.2.1 The Purpose 221
 - 10.2.2 Principal Aspects 221
 - 10.3 Evaluation 223
 - 10.4 Revision 224
 - 10.5 Learning 226
 - 10.5.1 Overfitting and Underfitting 228
 - 10.6 Learning to Fill and Modify Knowledge Containers 229
 - 10.6.1 The Vocabulary Container 229
 - 10.6.2 The Case Base Container 229
 - 10.6.3 The Similarity Container 234
 - 10.6.4 The Adaptation Container 238
 - 10.7 Applying Machine Learning Methods 238
 - 10.7.1 Regression Learning 238
 - 10.7.2 Artificial Neural Networks 239
 - 10.7.3 Genetic Algorithms 239
 - 10.7.4 Reinforcement Learning 241
 - 10.7.5 Clustering Methods 241
 - 10.7.6 Bayesian Learning 242
 - 10.8 Tools 242
 - 10.9 Chapter Summary 243
 - 10.10 Background Information 243
 - 10.11 Exercises 244
 - References 245
- 11 Development and Maintenance 247**
 - 11.1 About This Chapter 247
 - 11.2 General Aspects 247
 - 11.3 Development 248

- 11.3.1 Problem Formulation 248
- 11.3.2 Finding and Getting Data, Preprocessing 250
- 11.3.3 Case Acquisition 251
- 11.3.4 Prototypes and Evaluation 251
- 11.3.5 The Knowledge Containers 252
- 11.3.6 Which Additional Methods Can a CBR System Have? 254
- 11.3.7 Systematic Development of CBR Systems 254
- 11.3.8 Implementation Aspects 257
- 11.3.9 Combining CBR with Other Techniques 257
- 11.4 Maintenance 260
 - 11.4.1 Changed Environment and Techniques 261
 - 11.4.2 Maintenance and Knowledge Containers 263
 - 11.4.3 Systematic System Maintenance 266
- 11.5 Tools 270
- 11.6 Chapter Summary 271
- 11.7 Background Information 271
- 11.8 Exercises 272
- References 272

Part III Advanced Elements

- 12 Advanced CBR Elements 277**
 - 12.1 About This Chapter 277
 - 12.2 Discussion of the Relations Between Containers 277
 - 12.3 Contexts 281
 - 12.3.1 Generalities 281
 - 12.3.2 Different Contexts 282
 - 12.3.3 Contexts and Knowledge Containers 284
 - 12.4 Ontologies 285
 - 12.4.1 Ontologies in CBR 286
 - 12.5 CBR Systems 288
 - 12.5.1 Case Properties 289
 - 12.5.2 Case Base Properties 290
 - 12.5.3 Conditions 291
 - 12.5.4 Correctness and Provenance 293
 - 12.5.5 Distributed Case Bases 295
 - 12.6 Tools 296
 - 12.7 Chapter Summary 296
 - 12.8 Background Information 296
 - 12.9 Exercises 297
 - References 297
- 13 Advanced Similarity Topics 299**
 - 13.1 About This Chapter 299
 - 13.1.1 Foundations 299
 - 13.1.2 Formal Aspects 300

- 13.1.3 Meaning and Semantics 300
- 13.1.4 Subjectivity 303
- 13.1.5 Discussion of the Axioms for Similarity 306
- 13.1.6 First- and Second-Order Similarities 308
- 13.2 Miscellaneous Topics 310
 - 13.2.1 Nonfunctional Aspects 310
 - 13.2.2 Top-Down Versus Bottom-Up 310
 - 13.2.3 Jumps and Noise 312
- 13.3 Functional Dependency, Unknown and Redundant Values 312
 - 13.3.1 Functional Dependency 312
 - 13.3.2 Unknown Values 313
 - 13.3.3 Redundant Values 313
- 13.4 Additional Problems 314
 - 13.4.1 Similarity and Explanations 314
 - 13.4.2 Similarity and Logical Inference 314
- 13.5 The Knowledge Contained in the Measures 315
- 13.6 Tools 317
- 13.7 Chapter Summary 317
- 13.8 Background Information 317
- 13.9 Exercises 318
- References 318
- 14 Advanced Retrieval 321**
 - 14.1 About This Chapter 321
 - 14.2 General Aspects 321
 - 14.2.1 Case Retrieval Nets 321
 - 14.2.2 Fish and Shrink 326
 - 14.2.3 PROTOS: Another Two-Step Retrieval 330
 - 14.2.4 Fuzzy Retrieval 331
 - 14.2.5 Comparison 331
 - 14.2.6 Reducing the Search Space and Preprocessing 331
 - 14.3 Similarity Diversity 334
 - 14.4 Which Retrieval Method Should I Use? 336
 - 14.5 Tools 336
 - 14.6 Chapter Summary 336
 - 14.7 Background Information 337
 - 14.8 Exercises 337
 - References 338
- 15 Uncertainty 339**
 - 15.1 About This Chapter 339
 - 15.2 General Aspects 339
 - 15.3 Uncertainty Concepts and Methods 340
 - 15.3.1 Rough Sets 340
 - 15.3.2 Fuzzy Sets 342
 - 15.3.3 Basic Elements in Fuzzy Set Theory 344

- 15.4 Fuzzy Sets and CBR 348
 - 15.4.1 General Relations 348
 - 15.4.2 Fuzzy Cases 348
 - 15.4.3 Similarities 348
 - 15.4.4 Comparisons 350
- 15.5 Possibility, Necessity, and CBR 352
 - 15.5.1 General 352
- 15.6 Tools 354
- 15.7 Chapter Summary 354
- 15.8 Background Information 355
- 15.9 Exercises 355
- References 355
- 16 Probabilities 357**
 - 16.1 About This Chapter 357
 - 16.2 General Aspects 357
 - 16.2.1 From Probabilities to Measures 358
 - 16.2.2 From Similarities to Probabilities 363
 - 16.3 Bayesian Reasoning 365
 - 16.3.1 Dynamics 366
 - 16.3.2 Using the Nets 368
 - 16.3.3 Stochastic Processes 368
 - 16.4 Tools 370
 - 16.5 Chapter Summary 370
 - 16.6 Background Information 371
 - 16.7 Exercises 371
 - References 372
- Part IV Complex Knowledge Sources**
- 17 Textual CBR 375**
 - 17.1 About This Chapter 375
 - 17.2 General 375
 - 17.2.1 Text, Structure, and Levels 376
 - 17.2.2 Text Properties 380
 - 17.2.3 Problems in Understanding Text 382
 - 17.3 The Vocabulary Container 383
 - 17.3.1 Text Processing 384
 - 17.3.2 N-Grams 386
 - 17.3.3 Bag of Words 386
 - 17.3.4 Vector Representations 387
 - 17.3.5 Distributed and Reduced Representations 387
 - 17.3.6 Other Representations 391
 - 17.3.7 Identifying and Enhancing the Vocabulary 391
 - 17.4 The Case Base Container 395
 - 17.4.1 Hypertext 395
 - 17.4.2 Information Extraction 396

- 17.4.3 Information Entities in Basic Case Retrieval Nets 397
- 17.5 The Similarity Container 397
 - 17.5.1 Relevance-Oriented Measures 397
 - 17.5.2 Structure-Oriented Similarity Measures 398
 - 17.5.3 Measures for Segments 400
 - 17.5.4 Text to Text Similarity Measures 401
- 17.6 The Adaptation Container 402
- 17.7 What Textual CBR Method Should I Use? 403
- 17.8 Tools 404
- 17.9 Chapter Summary 405
- 17.10 Background Information 405
- 17.11 Exercises 406
- References 407
- 18 Images 411**
 - 18.1 About This Chapter 411
 - 18.2 General Aspects 411
 - 18.3 Image Structure 413
 - 18.3.1 Image Levels 413
 - 18.4 The Level Structure 415
 - 18.5 The Image Pixel Level 417
 - 18.5.1 The Geometric Level 418
 - 18.5.2 The Symbolic and Domain-Specific Level 421
 - 18.5.3 The Overall Level 423
 - 18.6 Semantics 425
 - 18.6.1 Aesthetics 428
 - 18.7 Knowledge Containers 431
 - 18.7.1 The Vocabulary Container 431
 - 18.7.2 The Similarity Container 431
 - 18.7.3 The Case Base Container 433
 - 18.7.4 The Adaptation Container 436
 - 18.8 Revision 436
 - 18.9 Standards 436
 - 18.10 How to Design Such a CBR System? 437
 - 18.11 Applications 437
 - 18.12 Tools 438
 - 18.13 Chapter Summary 438
 - 18.14 Background Information 439
 - 18.15 Exercises 440
 - References 441
- 19 Sensor Data and Speech 443**
 - 19.1 About This Chapter 443
 - 19.2 General Aspects 443
 - 19.3 The Level Structure 445
 - 19.3.1 The Low Level of Signals 446

- 19.3.2 The Feature Level 447
- 19.3.3 The Symbolic and the Overall Level 448
- 19.4 Semantics 449
- 19.5 Remarks on CBR Process Steps 449
 - 19.5.1 Cases and Case Bases 449
 - 19.5.2 Similarity 450
 - 19.5.3 Retrieval 451
- 19.6 Speech Recognition 451
 - 19.6.1 The Whole Speech Process 452
 - 19.6.2 The Role of the Levels 453
 - 19.6.3 Structure of a Speech Recognition System 454
 - 19.6.4 Cases and Case Bases 454
 - 19.6.5 Speech Feature Extraction and Similarity 454
 - 19.6.6 The Word Level and the Vocabulary in Speech 456
 - 19.6.7 The Overall Level 456
 - 19.6.8 Semantics 457
 - 19.6.9 Adaptation 457
 - 19.6.10 Errors 458
- 19.7 Applications 459
 - 19.7.1 Diagnosis 459
 - 19.7.2 Speech Applications 459
 - 19.7.3 Conversational CBR 460
- 19.8 Tools 460
- 19.9 Chapter Summary 461
- 19.10 Background Information 461
- 19.11 Exercises 462
- References 463
- 20 Conversational CBR 465**
 - 20.1 About This Chapter 465
 - 20.2 General Aspects 465
 - 20.3 Conversational CBR 466
 - 20.4 Knowledge Containers 467
 - 20.4.1 The Vocabulary Container 467
 - 20.4.2 The Case Base Container 469
 - 20.4.3 The Similarity Container 469
 - 20.4.4 The Adaptation Container 469
 - 20.5 Basic Conversation Systems 470
 - 20.5.1 Processing the Initial Description 470
 - 20.5.2 Dialogue Management 470
 - 20.5.3 Dialogue Formalisms 472
 - 20.5.4 Lengths of Conversations 475
 - 20.6 Architectures for Dialogues 476
 - 20.7 Quality and Evaluations of Dialogues 478
 - 20.8 More on Dialogues 479

- 20.8.1 The Role of Thesauri and Ontologies 479
- 20.8.2 Images in Dialogues 480
- 20.8.3 Dialogue Case Bases 481
- 20.9 Domains, Applications, Commercial Use 481
- 20.10 Tools 482
- 20.11 Chapter Summary 482
- 20.12 Background Information 483
- 20.13 Exercises 484
- References 484
- 21 Knowledge Management 487**
 - 21.1 About This Chapter 487
 - 21.2 General Aspects 487
 - 21.3 Knowledge Management 488
 - 21.3.1 Knowledge and Knowledge Management 488
 - 21.3.2 Knowledge and Decision Making 488
 - 21.3.3 Some Knowledge Management Problems 489
 - 21.3.4 Knowledge Management: An Organisational Discipline 489
 - 21.3.5 Knowledge Management Cycle 491
 - 21.4 Case-Based Reasoning and Knowledge Management 491
 - 21.5 CBR Implementing KM Cycles 493
 - 21.5.1 Knowledge Infrastructure and Organisation 493
 - 21.5.2 Knowledge Organisation and Retrieval 494
 - 21.5.3 Knowledge Retrieval and Reuse 495
 - 21.5.4 Knowledge Sharing 497
 - 21.6 For Which KM Tasks Should I Use CBR? 499
 - 21.7 Tools 500
 - 21.8 Chapter Summary 501
 - 21.9 Background Information 501
 - 21.10 Exercises 503
 - References 503
- Part V Additions**
- 22 Basic Formal Definitions and Methods 509**
 - 22.1 About This Chapter 509
 - 22.2 General Aspects 509
 - 22.3 Correctness and Logic 510
 - 22.3.1 Propositional Logic 510
 - 22.3.2 Predicate Logic 511
 - 22.3.3 Constraints 514
 - 22.3.4 Rules 514
 - 22.3.5 Reasoning in Logic 515
 - 22.4 Information Theory and Entropy 516
 - 22.5 Utilities 518
 - 22.5.1 Optimization 519

- 22.6 Chapter Summary 520
- 22.7 Background Information 520
- References 521
- 23 Relations and Comparisons with Other Techniques 523**
- 23.1 About This Chapter 523
- 23.2 General Aspects 523
- 23.3 Systems with Retrieval Engines 524
 - 23.3.1 Database Management Systems 524
 - 23.3.2 Information Retrieval Systems 525
 - 23.3.3 Pattern Recognition Systems 527
 - 23.3.4 Knowledge Comparison for Retrieval Systems 529
- 23.4 Explicit and Implicit Knowledge Representation 530
 - 23.4.1 Knowledge-Based Systems 530
 - 23.4.2 Machine Learning 531
- 23.5 Influence Factors 532
 - 23.5.1 Cognitive Science 533
 - 23.5.2 Analogical Reasoning 534
 - 23.5.3 Uncertainty 535
- 23.6 Chapter Summary 537
- 23.7 Background Information 537
- References 537
- Index 539**

Part I

Basics

Part I introduces the basic elements of Case-Based Reasoning (CBR) in such a way that no previous knowledge about the topic is required. The intended reader is anyone interested in learning about CBR. Part I provides a good basis for anyone who will use, design, modify, or make decisions about CBR systems. The overall audience is very broad and includes, among others, engineers, librarians, medical researchers, entrepreneurs, etc.

Chapter 1 provides introductory foundations for readers and an overview.

Chapter 2 provides a sound description of the methodology.

It starts by explaining that CBR is intended to reuse previous experiences. These experiences are the cases that are in principle pairs of the form (problem, solution). The reasoning says how to make use of the experiences. For a new problem one searches for an experience that has a problem which is closely related to the new problem. This reasoning takes place in a systematic way in the form of a process model. The knowledge needed for performing the processes is stored in a systematic way in boxes called knowledge containers.

We will explain two essential points for CBR:

- How to make use of an experience even if it differs from the actual problem.
- How to find a useful experience.

As a central concept for the second point is similarity that allows reasoning even from experiences where their situations differ from the actual one.

In Chap. 3 on the extended view the similarity relates problems and solutions directly without going back to experiences. This is frequently used in e-commerce where the problem is a demand that is directly related to a product. This extended to more general situations. Fortunately, almost all techniques developed for using experiences apply here as well.

Chapter 4 gives examples that support and complement the understanding of the previous chapters. It presents quite a number of examples. For these examples mostly the problem and its representation form for applying CBR is shown. The intention is to demonstrate the broad scope of CBR applications to the reader. There is

almost no attempt to show the specific solution approach. This can only be done on the basis of the material given in the subsequent parts.

However, it is strongly recommended that the reader frequently consults these applications.

It is our understanding that these chapters' contents should be used as parts of courses where merely an introduction to CBR is given.

Chapter 1

Introduction

1.1 About This Chapter

This chapter introduces this book on case-based reasoning (CBR); it is hence recommended to all readers. No previous knowledge about CBR is needed; only an interest in learning about it. It explains what CBR is and presents the main highlights in its history. This chapter aims to explain how to better use this book. The actual contents start in Chap. 2, Basic CBR Elements.

1.2 General Aspects

Case-based reasoning (CBR henceforth) has been a flourishing field for over three decades. It attracts researchers and entrepreneurs as well as practitioners. In fact, many practical applications have been commercially successful. There are several reasons for CBR being of interest; a few examples are:

- The origin of CBR as well as its relevance fall at the intersection of several disciplines of rather heterogeneous nature. They are mainly Cognitive Science; Computer Science and Computer Systems Analysis; Business; Library and Information Science; Engineering; and Education. Each of these disciplines has its own roots, its own methods, and its own foundations. This broadness opens the door for plentiful and diverse applications.
- The CBR methodology provides a computational model that is very close to human reasoning. CBR is rather intuitive; it is easy to understand. Consequently, when implemented, it utilizes a human paradigm in a computational context; benefiting from vast memory and speed provided by computers.
- CBR is able to deal with informal questions. An extensive and complex formalization of the problems is not required for CBR to be used.

Despite its intuitive plausible ideas, CBR can be technically complex. This book tries to bring you to a level at which CBR methods are understandable.

Fig. 1.1 Importance of learning from the past



In order to understand the CBR methodology, we invite you to think of simple, repeating problems one encounters on a daily basis, problems such as those of finding a spot to park a car, preparing a meal or finding a restaurant, suggesting strategies to the coach during a game, and interpreting people’s actions or words. Now we invite you to think about how often you have simply thought of a previous time you faced the same problem and ended up simply adapting a previous solution. These solutions can be as simple as following someone on foot in a parking lot in order to find a spot or going to the same restaurant you went to the previous week but remembering this time to ask for the dressing on the side, or even understanding your spouse by thinking “last time she said no, she actually meant yes!”

Many people have no real impression of what CBR may be concerned with. One reason is that the term *case-based reasoning* is not immediately understood in conversations everyday and even many technologists may have a problem grasping how to use a case for the intended reasoning. A simple way to make this clear is to refer to the use of analogy in law, in particular, in Common law.¹ A case is an event, for instance, a criminal event that occurred in the past and where someone got, say, a penalty as depicted in Fig. 1.1. If now a new case occurs, one looks for a case from the past that seems to be similar to the new one. Then one looks at the decision of the past case and tries to imitate it as closely as possible. This imitation is usually not a direct copying because the new situation is not totally identical to the old one. Therefore, some modification or adaptation has to take place: CBR includes the identification of the new problem, finding a similar one, recognising their differences, and adapting the old solution to solve the new problem. In CBR, such considerations have been extended widely and have been made accessible to be manipulated and executed computationally.

1.2.1 A Bit of the History

The history of CBR interleaves with the history of the development of models of memory. Cognitive Science is the field in which memory models are studied and cat-

¹Common law is the system widely used in English-speaking countries such as England and the US that is mostly based on precedents.

egorized. Associated with CBR are schema-oriented memory models, which have a long tradition and go back at least to Bartlett (1932).

Building upon schema-oriented memory models, Schank proposed the dynamic memory theory, or MOPs model of dynamic memory (Schank 1982). The dynamic memory uses a unit of representation, the memory organisation packet (MOP)—a dynamic structure used to represent patterns of situations in memory.

MOPs are stereotypical situations that tend to repeat. A computer program can become able to interpret situations if it knows the basic elements of such situations and can adapt. Examples of stereotypical situations are activities such as going on a trip, eating at a restaurant, going to class, purchasing a new car, and so on. These situations are characterised by actors, events, goals, scenes, certain instance types, and abstraction types.

Schema-oriented memory models and, in particular, MOPs had quite an impact on CBR. The schema-oriented memory models characterise the first period in the history of CBR. Systems like CYRUS, CHEF, and MEDIATOR, among others (see Riesbeck and Schank 1989 and Kolodner 1993), characterise this period. Many of the applications in this period are proofs of concept for a number of novel tasks for artificial intelligence (AI). They are also discussed in Chap. 23, Relations and Comparisons with Other Techniques.

Rather than being systematic engineering products, these systems are based on the kinds of complex tasks they can achieve and the lines of research they can motivate. The first workshops where research on CBR was presented took place in the US in 1988 (Kolodner 1988), 1989 (Hammond 1989), and 1991 (Bareiss 1991).

In 1992, the first German workshop on CBR was organised, and it became a landmark of the beginning of the second period: An era of knowledge compiled into engineering methods that allowed the development of CBR systems much more systematically and reduced development times and costs by a large degree. The INRECA projects (see Bergmann 2001, 2002) were the main applications promoting this shift. This second period also marked a correspondingly stronger European influence. Europeans started to organize workshops periodically. Workshops in Germany and the UK were now held annually. In the US, multiple workshops were organised with the main American AI conference: AAAI. The community produced two influential publications: In 1993, Janet Kolodner published *Case-Based Reasoning* (Kolodner 1993), a thorough book describing all the work in the field done thus far; and in 1994, Agnar Aamodt and Enric Plaza published an article on CBR foundations where they introduced the CBR cycle (Aamodt and Plaza 1994). The CBR cycle is widely recognised as the CBR methodology process model. By 1995, an international conference on CBR had started to be held every two years. An example of an influential and representative application of this era is Cassiopee—a CBR system developed in the INRECA project for the diagnosis and troubleshooting of CFM56-3 engines in the Boeing 737 aircraft.

The greater attention attracted by an international conference influenced the community; the result was a third period during which CBR publications started to appear in multiple conferences and journals. By the end of the 1990s, maintenance

was a new topic of interest; adaptation and retrieval continued to dominate. New areas starting to appear were conversational CBR, textual CBR, and knowledge management. A special track on CBR, first organised in 2001, became constant at the Florida Artificial Intelligence Research Society Conference: FLAIRS. Since 2002, the annual German workshop has been called the Workshop on Experience Management. This period is characterised by the use of CBR for more general methods and problems, which can be regarded as *experience mining*. This view of the CBR methodology extends the reuse of previous cases.

The third period started in the second half of the 1990s; by the mid-2000s the scenario had changed again. Recommender systems and diversity, which had become important topics in CBR, and data mining algorithms started to appear more often in CBR publications. This period may be seen as a transition to a stronger statistical influence, following the trends in AI in general. This period was well documented in a special issue called the Case-Based Reasoning Commentaries (Aha and Marling 2005). The issue included articles from the main research areas of CBR. This was the result of a community-wide effort that started in 2003 in a workshop in New Zealand with the community voting for the most significant articles in the field. The voting led to the identification of a set of areas and the writing of the commentaries.

This current period, since the commentaries, is characterised by the Web, collaborative and social systems, reduced representations, provenance, explanations and context-aware systems. Greene, Smyth, and Cunningham presented an analysis of CBR research themes based on keywords that identifies core publications in the field (Greene et al. 2008).

1.3 Intended Readers

The main readers this book targets are those who are potentially interested in applying CBR to a real problem. These are mainly students in universities. Given the strongest characterisation of CBR as a methodology, the main expected audience for this book consists of undergraduate computer science students from the third year on. Nevertheless, graduate students as well as students in programs other than Computer Science can also use this book. Case-based reasoning and related disciplines such as machine learning should become a regular topic in a computer science curriculum.

Today, computer scientists are no longer dealing with software development alone. They have to work in collaboration with systems analysts, information technologists, software engineers, and other types of engineers. In fact, any technology or engineering program would benefit from including a course in CBR.

Also part of software development projects are business, marketing, and financial analysts. Furthermore, the close relationship between CBR and problem solving and decision making makes it an attractive discipline for business and management programs and Master's in Business Administration. The affinities with knowledge management, discussed in Chap. 21, Knowledge Management, corroborate the relevance to organizational disciplines and substantiate its potential benefit to Library

and Information Science. In fact, given their expertise in user behaviours, it has become more common that library scientists are a part of software development projects. Consequently, we also suggest this book be used in courses in Library and Information Science.

The students who will benefit from this book can come from different departments and can have different application interests. The large number of applications in areas such as medicine and law justify the offer of an elective CBR course in these programs. In this respect, this book will support better interdisciplinary education.

The audience also includes lecturers who want to teach such courses for their (not uniform) audience. For this reason, we want to provide to the lecturer a certain degree of freedom. Using the book the lecturer gets advice on what is basic and what is advanced. Each chapter starts with a recommendation and tries to identify for which kinds of students it may be useful. The contents of Parts III and IV are very much dependent on the types of applications and the special interests of the students. Hence, the instructor can plan a course depending on the course goals, specific program, and teaching objectives. For the support of the lecturer, we include exercises at the end of each chapter, which can be used as the starting points for further projects.

Finally, this book can be used in conjunction with additional and specific material for adjacent topics and disciplines where case-based problem solving is common. An example of an adjacent discipline would be machine learning, where some additional material on data-intensive learning methods should be aggregated. Courses on recommender systems and e-commerce systems would be two other courses in which this book could be used as a substantial part of the curriculum. Those other courses could be on artificial intelligence for nursing, library management, medicine, law, and so on. CBR's purpose in this context is to help solve problems in different disciplines. These would be taught in higher-level courses where the additional material should come from classic and recent publications in interdisciplinary problems.

In Fig. 1.2, we give an overview and dependencies of the chapters.

1.4 Outline of the Book

The book is organised into four parts. Each part contains a short preliminary section that gives an overview. The diagram in Fig. 1.2 gives a guide to the reader and the lecturer. The arrows indicate the order in which it is suggested that you read the chapters. Here we give a brief introduction to each of the chapters in the book.

1.4.1 Part I: Basics

Part I consists of basic elements for understanding, designing, implementing, and deciding about the successful use of the CBR methodology for some purpose.

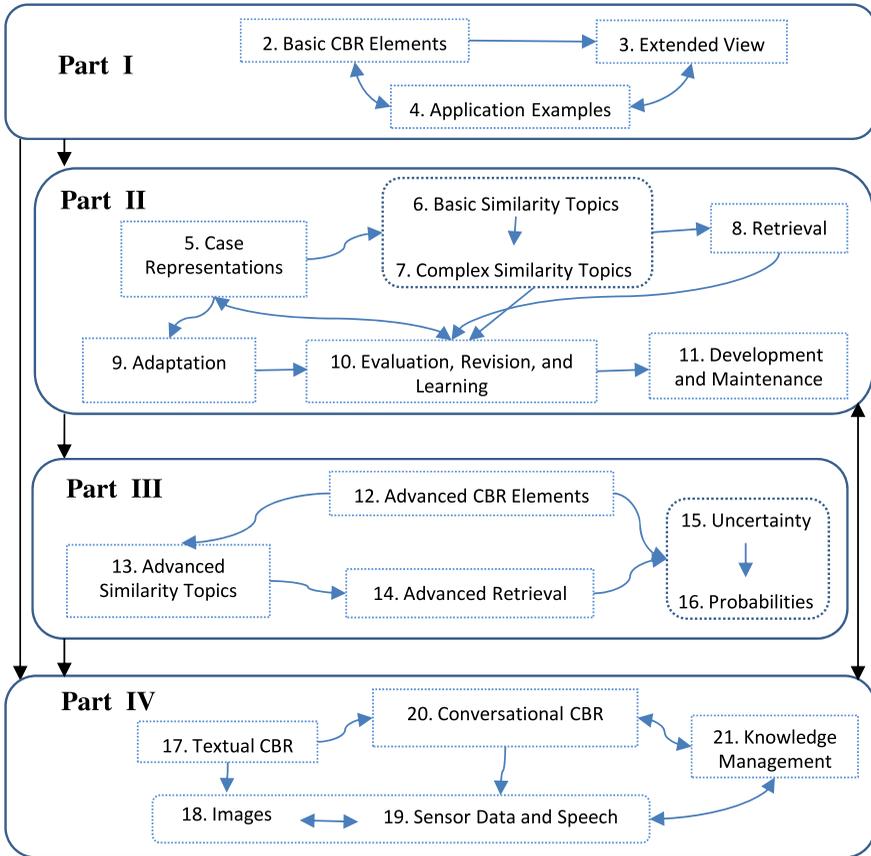


Fig. 1.2 Directions for recommended reading

1.4.1.1 Chapter 1: Introduction

This very chapter; it is an introduction to the book, including the history of CBR.

1.4.1.2 Chapter 2: Basic CBR Elements

This chapter contains the basic notions that are necessary to understand the rest of the book. Concepts in this chapter are given in an informal way and will be extended and possibly formalized in later chapters.

These initial concepts are required for understanding the CBR methodology. The concepts are then put into context with two major models of the CBR methodology.

- A process model based on the Aamodt-Plaza CBR cycle (Aamodt and Plaza 1994). This explains the major steps applied in every application. The process model includes problem formulation, retrieval, reuse, revision, and learning.

- A knowledge model based on Richter's knowledge containers (Richter 1995). These are description elements of CBR systems. The containers are the vocabulary, the case base, the similarity measure, and the rules.

These processes and containers are explained in the chapters in Part II.

1.4.1.3 Chapter 3: Extended View

This chapter presents a form of CBR that does not compare previous problems with a new one but rather compares problems with solutions directly. This avoids the use of experiences stored as cases. Two major application areas are e-commerce and recommender systems.

1.4.1.4 Chapter 4: Application Examples

This chapter contains a variety of application types and concrete applications. The applications are described in terms of their problem and how they are formulated. We describe the solution approach but do not delve into too many technical details. The intention is that the reader should get a first idea about the wide scope of CBR applications, and about how CBR works and when it is recommended. This chapter can be read at any point although we advise you do it after reading Chap. 2, Basic CBR Elements.

1.4.2 Part II: Core Methods

Part II contains the core methods. It serves anyone who wishes to have a sound understanding of CBR by strengthening and consolidating the understanding of the concepts introduced in Part I.

1.4.2.1 Chapter 5: Case Representations

This chapter discusses the various ways of formally representing cases. It starts with flat attribute-value descriptions and goes then to more complex descriptions such as object-oriented methods. Indirect descriptions that use links or references are also discussed. The local-global principle for building complex structures is introduced.

1.4.2.2 Chapter 6: Basic Similarity Topics

This chapter is the first of two chapters on similarity in Part II (Chaps. 6 and 7). The need for two chapters stems from the importance of similarity as a central concept in CBR. This first chapter presents the basic concepts and a large number of

similarity measures for flat attribute-value vectors. We classify the measures into seven different categories. We discuss the use of the local-global principle for building measures. We discuss how “more similar” means “more useful” and illustrate this with several typical local measures. We pay special attention to the roles of attribute dependency and weights for linear measures, which are closely related. For dependencies we investigate the use of new (virtual) attributes.

1.4.2.3 Chapter 7: Complex Similarity Topics

This chapter is the second similarity chapter. It starts with measures on complex object descriptions. These include graphs, taxonomies, and general object-oriented representations. We then expand to measures for processes, workflows, and time series.

1.4.2.4 Chapter 8: Retrieval

This chapter deals with the process model step of retrieving a useful case from the case base. Technically, these are the k -nearest neighbours resulting from the chosen similarity measure. We introduce the basic retrieval techniques. This chapter starts by using simple data structures, including geometric structures and Voronoi diagrams, followed by index-oriented retrieval. These methods are suitable for cases organised in trees; thus, we discuss kd-trees and decision trees.

1.4.2.5 Chapter 9: Adaptation

This chapter is devoted to the reuse step from the process model. After the most similar (useful) candidate case is retrieved, its solution is to be suggested for reuse to solve the new query problem. The issue with reuse is that the obtained solution may not be used as it is. The reason is that the old and the new situations, despite being similar, are likely not identical. The necessary modification of the solution is called adaptation. The possible adaptations are usually defined by a finite set of rules. There are multiple techniques for chains of adaptations and they need to be planned, as sometimes one step is not enough to create a valid solution.

1.4.2.6 Chapter 10: Evaluation, Revision, and Learning

This chapter presents the following two steps in the CBR process model: revise and retain. Revision includes an evaluation of the suggested solution, and the revision step per se that attempts to correct a poor evaluation. Learning refers to the retain step of the process model, where a successfully evaluated and novel solution can be stored as a new case.

The revise step differs from the reuse step in that reuse considers a new solution in a model only. In contrast, revise looks at the given solution from the viewpoint of reality (or at least a simulation of it). This requires a report on errors or evidence of weaknesses. The latter is done by evaluations. Revision takes place when a specific error is reported. Learning results in the improvement of the case base knowledge container. Both the CBR process model and knowledge containers are described in Chap. 2, Basic CBR Elements.

1.4.2.7 Chapter 11: Development and Maintenance

This chapter includes these two topics because they are closely related to each other. Development is closely related to maintenance in that similar tasks are required in both. Both are concerned with filling and structuring the knowledge containers. Both make use of the techniques presented in the previous chapters that deal with improving the containers. The main difference is that the first time one applies them is in the development phase; when they are applied later again, it is as part of maintenance.

1.4.3 Part III: Advanced Elements

Part III is advanced, and thus intended for those interested in understanding, designing, implementing, and deciding about more sophisticated applications, and those interested in research. It may not be required in undergraduate-level courses or in courses not on computer science.

1.4.3.1 Chapter 12: Advanced CBR Elements

This chapter extends Chap. 2, Basic CBR Elements, by including material that gives a deeper insight into CBR. Nevertheless, it may not be of interest to beginners. The first part deals with properties of cases and case bases that influence the behaviour of a CBR system and its relation to the user. Such properties deal with, among other things, numerical magnitudes like size or with provenance and correctness. As an extension, we discuss distributed case bases. The second part is devoted to a systematic study of contexts and ontologies. Contexts determine, among other things, the precise intentions of problems and solutions. Ontologies are used for systematically describing contexts.

1.4.3.2 Chapter 13: Advanced Similarity Topics

This chapter provides mainly additional and background information for the previous similarity chapters (Chaps. 6 and 7). It starts with formal aspects of the semantics of similarity measures. We present a formal foundation based on utility.

Another topic is subjectivity; this is not the same as vagueness. We revisit popular axioms for similarity, which leads to the question of what knowledge the measures can contain. This subsumes, among other things, explanations in CBR and relations to logic. Also, the discussion considers what one can do if the case representation has defects.

1.4.3.3 Chapter 14: Advanced Retrieval

This chapter extends Chap. 8, Retrieval. We introduce advanced methods such as case retrieval nets, fish and shrink, and fuzzy retrieval. We discuss the use of footprints for the reduction of the search space. We also discuss the problem known as diversity.

1.4.3.4 Chapter 15: Uncertainty

This chapter deals with general aspects of uncertainty and with fuzzy sets. We present the different kinds of uncertainty and the close relations between CBR, fuzzy sets and possibility theory.

1.4.3.5 Chapter 16: Probabilities

This chapter is concerned with CBR's relations to probability. Probabilities give rise to similarity measures. This includes similarity measures between random variables. We also discuss Bayesian reasoning.

1.4.4 Part IV: Complex Knowledge Sources

Part IV focuses on areas within CBR that gather specific application types. These require individual chapters because they deal with complex knowledge sources. Text is difficult to understand, conversations are difficult to undertake; images and speech are hard to capture and represent; knowledge management is an organisational problem. This part is intended for both basic and advanced readers. For using all of these representation types we use a level structure. The lowest level considers how information is stored in the computer and the highest level deals with its meaning.

1.4.4.1 Chapter 17: Textual CBR

This chapter extends the use of CBR to problem situations where knowledge is originally available in textual form. The chapter reviews the main problems with understanding from textual sources and presents some of the techniques developed to represent text for retrieval and reasoning purposes. Techniques used mostly in the vocabulary and similarity containers are included.

1.4.4.2 Chapter 18: Images

This chapter discusses another example where the basic entries to a complex system are not attributes. In CBR they may occur in queries as well as in solutions. This is due to the fact that images can contain important information. In the past, they were mainly investigated in image processing, which is not a topic here. Less attention was given to understanding images than is needed for queries as well as solutions. The level hierarchy starts at the pixel and goes up to where the overall understanding is located. In the examples we mainly refer to medical images.

1.4.4.3 Chapter 19: Sensor Data and Speech

This chapter includes elements of sensor data and data coming from spoken speech. We again use a holistic approach, starting from elementary signals up to the overall meaning. CBR does not only occur on all levels in queries and solutions: classical speech recognition is a CBR application itself. On the level of signals there are other technical measurements. These contain information too. For this reason, several techniques are presented jointly for measurements and speech.

1.4.4.4 Chapter 20: Conversational CBR

This chapter focuses on the use of dialogues to assess the query problem from a user. The chapter discusses the use and the design of a conversational CBR system. The main technical aspects relate to dialogue management and evaluation.

1.4.4.5 Chapter 21: Knowledge Management

This chapter discusses the many affinities and overlapping concepts in CBR and Knowledge Management (KM). We discuss how CBR can be beneficial to KM and illustrate this by explaining applications.

1.4.5 Part V: Additions

These chapters focus on fields outside CBR that share strong implications. They are intended for all readers with interests in any of those fields. These are additions given their independence with respect to the rest of the book.

1.4.5.1 Chapter 22: Basic Formal Definitions and Methods

This chapter provides formal foundations for several concepts and techniques discussed in Chap. 1. In particular, the terminology used in the book is given. The major topics are knowledge representation with an emphasis on logic, entropy, partial orderings, utility, and optimization.

1.4.5.2 Chapter 23: Relations and Comparisons with Other Techniques

In this chapter we compare CBR with other system types. We distinguish three categories: systems in competition with CBR, systems that can cooperate with CBR, and the roots of CBR. Examples for these categories are database systems, machine learning methods, and cognitive sciences.

1.4.6 Chapter Structure

Apart from this introductory chapter, all other chapters consistently include sections with purposes we now explain.

1.4.6.1 About This Chapter

Each chapter starts with a brief examination of its intended audience, whether previous knowledge is needed, and its purposes.

1.4.6.2 Tools

These are intended to mention programs that may be available for testing or implementing the methods and techniques discussed in the chapter. They provide informal links to and comments on tools useful for the content of the chapter. This section is especially important for those who want to build their own CBR system.

The exception is Chap. 4, Application Examples, where no content other than examples is presented.

1.4.6.3 Chapter Summary

A summary of the chapter is given, and its main concepts and ideas are highlighted. This allows the reader to get the big picture of the chapter. It also helps instructors in identifying the most important aspects.

1.4.6.4 Background Information

This section includes remarks on the history of the main concepts covered in the chapter, connections to other chapters of the book and to other areas, further readings and other aspects specific to its contents. The main text in the chapters does not include references, for simplicity. Therefore, it is in the background information section that we indicate proper references, and discuss a bit of history and angles that may not be of great technical importance but may interest some of our readers.

Chapter 1 does not have a background information section because it is of background nature. Necessary references are provided in its text.

1.4.6.5 Exercises

All chapters include exercises. The purpose of these exercises is to provide suggestions so instructors can use them for teaching. These vary substantially in nature in order to make available exercises for students from different backgrounds. Furthermore, the exercises can sometimes expand to projects that are more ambitious.

1.4.6.6 References

Chapters conclude with sections listing the references and bibliography. They are meant to be consulted as background and also as further reading.

References

- Aamodt A, Plaza E (1994) Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Commun* 7(1):39–59
- Aha DW, Marling C (eds) (2005) Special issue on case-based reasoning commentaries. *Knowl Eng Rev* 20(3):201–328
- Bareiss R (ed) (1991) Proceedings of a workshop on case-based reasoning, Washington, DC, 1991. Morgan Kaufmann, San Mateo
- Bartlett FC (1932) *Remembering: a study in experimental and social psychology*. Cambridge University Press, UK
- Bergmann R (2001) Highlights of the European INRECA projects. In: Aha DW, Watson ID (eds) *ICCBR 2001: case-based reasoning research and development*. 4th international conference on case-based reasoning, Vancouver, BC, Canada, July/August 2001. Lecture notes in computer science (lecture notes in artificial intelligence), vol 2080. Springer, Berlin, p 1
- Bergmann R (2002) *Experience management: foundations, development methodology, and internet-based applications*. Springer, New York
- Greene F, Smyth B, Cunningham P (2008) An analysis of research themes in the CBR conference literature. In: Althoff K-D et al (eds) *ECCBR 2008: advances in case-based reasoning*. 9th European conference, Trier, Germany, September 2008. Lecture notes in computer science (lecture notes in artificial intelligence), vol 5239. Springer, Berlin, p 18
- Hammond K (ed) (1989) Proceedings of a workshop on case-based reasoning, Pensacola Beach, FL, 1989. Morgan Kaufmann, San Mateo

- Kolodner JL (ed) (1988) Proceedings of a workshop on case-based reasoning, Clearwater Beach, FL, 1988. Morgan Kaufmann, San Mateo
- Kolodner JL (1993) Case-based reasoning. Morgan Kaufmann, San Mateo
- Richter MM (1995) The knowledge contained in similarity measures. In: Keynote at ICCBR-95: 1st international conference on case-based reasoning, Sesimbra, Portugal, October 1995
- Riesbeck CK, Schank RC (1989) Inside case-based reasoning. Erlbaum, Hillsdale
- Schank RC (1982) Dynamic memory: a theory of reminding and learning in computers and people. Cambridge University Press, New York

Chapter 2

Basic CBR Elements

2.1 About This Chapter

This chapter introduces the basic concepts of case-based reasoning (CBR). It does not require any previous knowledge about the topic. The intention is that the reader should understand the principal ideas and follow the descriptions of the first examples, the remaining chapters in Part I, and Part II. This chapter provides the fundamental basis for understanding the remainder of the book. In most cases no formal definitions are used. In later chapters the concepts will be extended and put on a more formal basis, and they will be illustrated by examples.

2.2 General Aspects

Case-based reasoning (CBR) is a methodology for solving problems. These problems may be of a variety of natures. In principle, no problem type is excluded from being solved with the CBR methodology. The problem types range from exact sciences to mundane tasks. However, this does not mean that CBR is recommended for all problems. Throughout this book, we will give examples of problems and explain how CBR can be used. As a result of understanding these circumstances, the reader will develop a sense of when CBR is recommended.

Because CBR is essentially based on experiences, this chapter will discuss the main aspects of the CBR methodology and how it uses experiences in a specific way to solve problems. It is inherent in using and reusing experiences that they embed answers to problems or ways to get solutions. These answers can, for instance, help to solve difficult combinatorial problems, as an add-on they can also suggest or improve solutions where uncertainty is involved.

2.3 Case-Based Reasoning

The term case-based reasoning consists of three words and they need a short explanation. A **case** is basically an experience of a solved problem. This can be repre-

Table 2.1 Two recorded experiences of the same event

Experience report 1. April 10	Experience report 2. April 10
10.45 Problem reported	Morning: Strange loud noise, dust came out, speed of machine is slowing down
11.00 Maintenance arrived	10.45 Machine stopped
11.50 Expert from group C was called	11.15 Test 35 and test 45 failed
12.10 Expert arrived	12.15 Pressure was detected not working
13.15 Exchange part arrived	12.25 Valve A was working improperly, after exchange the problem was solved
14.15 Part was built in	
14.30 Machine is running	

sented in many different ways. A case base is a collection of such cases. The term **based** means that the reasoning is based on cases, that is, cases are the first source for reasoning. The term most characteristic of the approach is **reasoning**. It means that the approach is intended to draw conclusions using cases, given a problem to be solved.

The kind of reasoning is, however, quite different from reasoning in databases and logic. The most important characteristic that distinguishes case-based reasoning from other kinds of reasoning is that it does not lead from true assumptions to true conclusions. This means that even if the solution in a recorded case were correct for its original problem, this may not be the case for a new problem. This possibility is based on the general fact that the situation in the recorded experience may not be exactly the same as that in the new problem. In fact, to be reused, it only has to be “similar”. Therefore, the result of making use (or reuse) of the experience may only be “close” to the correct solution of the new problem. This means that applying CBR is a kind of approximate reasoning. Consequently, in order to more precisely describe its nature, we will investigate the concepts of being *similar* and *close* in more detail. In fact, CBR is essentially centred on these terms and most parts of this book are meant to describe this form of reasoning.

2.4 Experiences and Cases

Experiences are essential for CBR. In general, an experience is a recorded episode that occurred in the past, such as “Remember, last year in Italy we had a similar problem with our car. The hint the mechanics gave us worked pretty well. We had this problem quite often; and our usual way for fixing the problem always worked somehow”. Such experiences are used to help solve future problems or make future decisions. However, not every recorded episode will be useful in this respect. For this reason, we consider two recorded experiences of the same event, in Table 2.1.

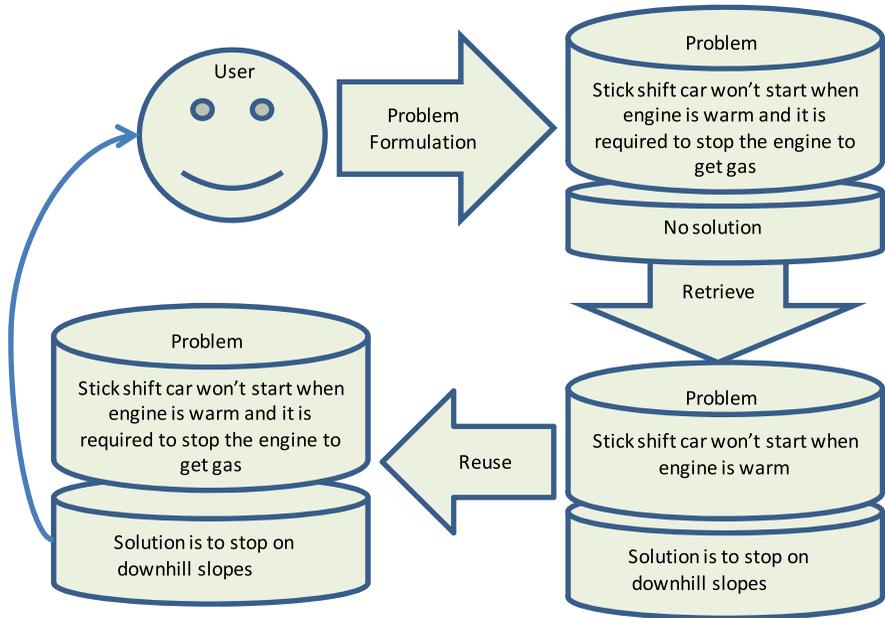


Fig. 2.1 Steps of the experience

Both experiences in Table 2.1 record what happened in the past. But the first one is almost useless for advice on fixing such problems in the future. It may, however, be used for administrative purposes. The second report offers some advice for a procedure that can be useful, although no detailed description is provided. This shows that past events can be viewed from many different perspectives.

To model a real-world situation there is no need to consider all aspects of a problem. The task is to identify those that are relevant and useful for solving the problem. In complex and unclear situations this may sometimes be difficult.

Suppose you go on a trip with your colleagues, driving your new used car, used but new to you—the cheapest you could afford. The car has been working fine, except that it will not start once the engine is warmed up. You are running out of gas and stopping at a gas station and waiting for the engine to cool down would delay your trip. As you share the problem with your colleagues, one of them is quickly reminded of a time when his cousin had a similar problem starting her car, so she would only park or stop the car on downhill slopes. Given that her car has a stick shift, she could simply let it ride to get it started. Because your car is also a stick shift, you can reuse the same solution, which is feasible because you are in a hilly area.

In Fig. 2.1 the diagram breaks down the CBR methodology into steps, which helps explain how CBR uses experiences to solve problems. These steps are executed around the concepts of problems and solutions. Problems, solutions, and these steps will cover a major part of this chapter.

2.4.1 Parts of a Case

Cases can be quite complex and consist, as mentioned, of whole stories. CBR uses them for solving problems; therefore, there must be something in the experience that talks about a problem and its solution. In a simple view, CBR divides an experience into two parts:

- A problem part (or a description of a problem situation).
- A solution part that describes how one has reacted.

Often one restricts CBR to solutions that have been successful, but that is by no means necessary or adequate. A failed solution is also an important piece of information that states what one has to *avoid*. The coexistence of both successful and failed experiences leads to the following definition.

Definition 2.1

- (a) Positive experiences (cases) implement successful solutions and lead to the advice: Do it again!
- (b) Negative experiences (cases) implement failed solutions and lead to the advice: Avoid this!

When positive and negative cases occur one can introduce two sets of cases: C^+ (positive) and C^- (negative) cases. Negative cases occur often in the context of decision making where one has to choose from different alternatives or when advice has to be given. Negative cases have to be distinguished from cases that contain errors.

Major types of experiences occur in:

- (a) Classification: Decide the class to which an object belongs. For instance, classify mushrooms into the two classes “edible” and “poisonous”.
- (b) Diagnosis: Decide what the diagnosis of a problem is. For instance, determine whether what causes a car to malfunction is lack of gas (see also the example given below).
- (c) Prediction: Decide what happens tomorrow. For instance, for predict expenses for a firm for a given month in a given year.
- (d) Planning: Decide on a sequence of actions to reach a given goal. For instance, make travel plans.
- (e) Configuration: Decide which elements to include. For instance, decide how to select technical features and components of equipment.

In Chap. 4, Application Examples, there are several problems in these categories given and it is shown how they can be approached by CBR. Chapter 4 defines and discusses the different forms of reasoning that are embedded in different experiences. Because experiences can perform different reasoning tasks, it is important that a CBR system be uniquely designed to tailor each type of experience. Consequently, one considers one type of problem at a time, that is, one reasoning task.

A CBR system is typically designed to perform one reasoning task. These systems offer an extended view of CBR.

Recall we mentioned that, to be reused, a recorded experience needs only to be similar to the new problem. This form of approximate reasoning generates an additional, though optional, component in cases. This third component is usually regarded as case outcome. Case outcomes do not retain knowledge about the experience itself, but they can be seen as a place to record meta-experiences, that is, information about uses of an experience. For example, how often a case is used, how successful it has been, and so on.

While humans can understand accounts of experiences told in everyday language, computers require some formality. Although natural to humans, the recognition of similarity and the consequent ability to reuse experiences requires an analogy when using a computer. This is a formal system that is intended to represent experiences so they can be reused.

Sometimes experiences are not given in such a suitable, formal way because they may rely on experiences that are informally described, for instance, in a textual form. Part IV is dedicated to situations such as when experiences are available in textual, visual, or conversational forms.

2.4.2 Problems

Problems are central to CBR because the main purpose of the methodology is problem solving. The formulation of a problem is sometimes difficult because it refers to the context in which it is stated. So, each problem formulation requires a different kind of solution. For example:

What is the price of this car?

- One answer could be: Too expensive for us.
- Another answer could be: \$252,600.

It is obvious that one has to know the context in which the problem is stated in order to find out which answer is appropriate. In other words, for a precise statement the context has to be included in the problem formulation.

Part of the context is often the inherited culture. Consider for instance the Roman and the Anglo-Saxon laws. In the Roman law there are rules that say that in such and such a situation the decision is in favour of the defendant. In the Anglo-Saxon law the decision is traditionally based on the relationship (i.e., analogy) between an event that occurred in the past and the actual event. This latter kind of decision making is what CBR applies.

Another cultural point is what is considered as important in planning. For instance, what counts more, building a street or a school? Depending on the culture, laws may be different in different areas. Other cultures are provided by different sciences such as medicine, business and engineering; even large companies have developed their own culture. The CBR context has to take this into account because

transferring solutions across cultures is problematic. For example, each bank has developed its own policy for giving loans to customers. The same bank may interpret the policy differently in each different country it operates; this becomes apparent during financial crises.

There are two types of problems we discuss in the context of the CBR methodology. The problems in the cases recorded as experiences are usually referred to as problems in CBR. The cases in the case base can sometimes be distinguished as candidate cases, as they are candidates for reuse. However, the entire CBR process is triggered by a problem. This is the new problem, or the actual problem that motivates a user to find a problem-solving method. To make this distinct from other uses, we henceforth refer to this as the query problem or, simply, the problem.

This section introduces problems and problem types, where the latter are more general. Next, we distinguish between a problem and a solution. These simple and intuitive notions are intended to eventually have formal definitions. Alternatively, we will use the term query instead of problem, and answer instead of solution.

2.4.3 Solution Types

The possible ways of representing a solution vary:

- It can be just a solution in the narrow sense.
- It can contain in addition:
 - Comments, illustrations, explanations.
 - Advice on how to use the solution.
 - The effect by describing what occurred with the solution in the past.
 - Remarks on the strategy with which the solution was obtained.

In simple cases the solution contains a name or simple data, for instance, an object or an expected temperature. It may also be a project with values given to predefined attributes, such as jogging three times a week for 45 minutes. Solutions may also have a complex object-oriented structure as a technical object. Even more complex are solutions for planning and those in textual or image form.

In a complex situation the solution is a decision for performing an action or even a process. Here one has to distinguish the decision from the action; the action refers to an implementation and run of a strategy that may change states of variables. While the decision is usually clearly formulated, the outcome of the action may be uncertain. Suppose, for instance, that we have the choice between the different lotteries L_1, \dots, L_n and we want to choose a lottery that has maximal expected win. Then our solution can only present us a certain lottery; the win is represented as a probability distribution. Hence the computed probability has to be mentioned in the solution description. Another example is if we decide to fly to Toronto. The execution may fail or be postponed because of various unforeseen events. The latter means that the result of using a solution is uncertain because of unexpected external results like bad weather or an earthquake. If these are likely to happen one should extend

Table 2.2 Attributes of the case

Attributes and their values	
Symptom	Unusual car noise
Observations	Knocking engine
Since last inspection (month)	3
Rhythmic pounding	No
Related to car speed	No
Oil pressure light flickering	Yes
Leaking oil	No

the solution by an entry “effect” for describing what really happened. The user who sees the solution does not know this. If it is added then the user may get a hint for some possible adaptation. Finally, there are situations where the usefulness of the solutions can only be judged if they are executed in reality. This is the case with decisions for organising city traffic, or, more generally, with making predictions.

2.5 Case Representations

Now we know that cases are experiences and that such experiences have a context. We also know that cases include problems and solutions. The next step in introducing the CBR methodology is to explain how a case is explicitly represented and how cases are organised. Note that most of the formalisms used are not new. In fact, they can be considered quite common and they are used in many other problem-solving methodologies.

2.5.1 How Cases Are Represented

The simplest way to represent a case is by using feature-value pairs. A feature value pair is used to represent a state of an entity, for example, colour of an entity, “Jessica’s car is red”, where the feature is the colour of the car and the value is red, and the entity is Jessica’s car. Instead of the word “feature” the word attribute is often used, and we will freely switch between these.

Features need to be identified for both problem and solution. Suppose someone has a headache and needs a diagnosis indicating what problem may be causing the headache. In Table 2.3, we find several cases for this.

A set of features has to be selected to represent cases. Each patient is represented in a case. Table 2.2 depicts one case with feature-value pairs for problem and solution.

Cases have to be described in some language. In principle, such a language can have an arbitrary character. This is only a preliminary view; more details and varia-

tions are given in Chap. 5, Case Representations. Feature value representations are, in fact, just an attribute-value vector.

Definition 2.2

- (i) For a given set U of objects, an attribute A assigns to each object $O \in U$ some value taken from a set $\text{dom}(A)$, the domain of A .
- (ii) An attribute-value description is a finite vector of attributes.

This means the represented object is just an attribute-value vector. The problems and the solutions are described in this way. It is, however, a very simple definition of a concept; it will be extended later in various ways; see Chap. 5, Case Representations.

The need for more complex representations originates from the fact that such a representation cannot entail everything we can see and that is of interest. Consider the example in which we want to describe a car failure in order to represent it as a case. A case description limits the scope from the potentially infinite properties of a car to only a small part. Out of the thousand parts a car may have, many are irrelevant to most problems. This leads to the question: Which attributes should one take? The question cannot be answered universally, not even for cars. The point is that the chosen attributes should be relevant for the problem type in question. Our early example was for diagnostic purposes, typical when a fault occurs. In order to sell a car however different attributes would be relevant; as we see in Table 2.2.

A case would be a description of the car problem together with a description of the solution. Here, a problem is just a case without a solution, in this example unusual car noises. Table 2.2 presents the example through its attributes. One must realise that there are numerous car failures that refer to many different aspects of a car. However, within a certain type of failure the diversity is rather restricted. Hence, one has either to know what the possibly relevant attributes are or where one can find them. Table 2.2 shows some attributes.

In order to make use of experiences for solving the problem of finding a diagnosis and a repair we need a collection of many experiences to choose from. These will be a collection (or set) of cases. As a general advice, we can look at humans describing a failure. That means we did not invent something new here. The CBR goal is just to support the usage of those experiences. Even if all attributes used are of interest, it is not guaranteed that they all have the same importance. This will be considered later when similarity is discussed.

2.6 Case Bases

A case base is a memory; it contains a collection of cases that is used in the context of the CBR methodology for the purpose of performing a reasoning task.

Definition 2.3 A *case base* is a collection of cases.

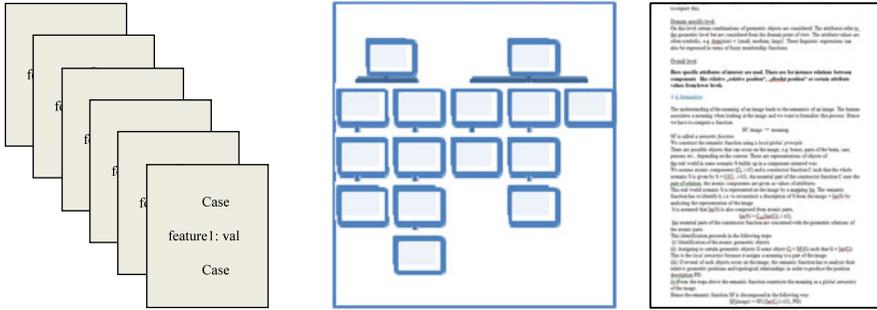


Fig. 2.2 Three types of case organisation: flat, structured, unstructured text

A case base is a data source and usually it is finite. What is *specific to CBR* is how a case base is used. In Chap. 23, Relations and Comparisons with Other Techniques, we contrast case bases with databases. The usage for CBR requires special ways of utilizing the case base. The word “memory” is heavily used in cognitive sciences too; this will also be discussed in Chap. 23.

2.6.1 How Are Cases Organised?

We have three main types of case organisation: flat, structured, and unstructured (e.g., text, images). Figure 2.2 illustrates the three basic types of case organisation. Note that these forms already suggest different programming paradigms, but we will only get into the programming aspects in Chap. 5, Case Representations.

2.6.1.1 Flat Organisation

The flat organisation is the simplest to design and implement, and most suitable for a small number of cases. Table 2.3 shows an example of cases in a flat organisation. Attributes are listed in Table 2.3 in the leftmost column. They are used for representing case problem and solution. The six cases are represented through different values. There are no relationships between the cases. No one case has any relationship to another that needs to be represented, which means that the representation is complete. This is an example of a case base, that is, a collection of six cases that can be used in the context of the CBR methodology to diagnose the potential source of a headache.

2.6.1.2 Structured and Unstructured Organisation

Cases can be organised into structures such as hierarchies and networks. To be structured, however, cases do not necessarily require a hierarchical organisation. The relationship between two cases will have specific characteristics. An object-oriented

Table 2.3 Six diagnosis cases

Attributes	Case id					
	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6
Nausea	Yes	Yes	Yes	No	No	Yes
Fever	Yes	No	No	No	No	No
Malaise	Dizzy	Dizzy	Dizzy	No	Listless	Listless
Blood pressure	Normal	Normal to low	High	Normal to high	Normal	Normal to high
Vision changes	No	Yes	No	No	No	No
Shortness of breath	No	No	Yes	No	No	No
Patient name	Bart	Marge	Lisa	Homer	Maggie	Ned Flanders
Diagnosis	Influenza	Migraine	Heart problem	Stress	Vitamin deficiency	Hangover

organisation is structured. The structured organisations can be beneficial when the number of cases is very large.

Cases are very commonly hidden within texts or images. At this point we will not yet discuss other concerns pertinent to unstructured organisations. We have entire chapters dedicated to them, namely, Chaps. 17, 18, and 19.

2.7 Similarity and Retrieval

The purpose of retrieval is to retrieve the case from the case base (i.e., a candidate case) that is so similar to a given new problem that their solutions can be swapped. One of the implications of this concept of similarity within the CBR methodology is that CBR's similarity is not a general concept, but a polymorphic concept that varies for each case base. One can, for instance, use the same case base with different measures for different purposes.

The purpose of any problem-solving method is to obtain a good solution, ideally even the best. The meaning of this is given by the user. Now that we understand that the CBR methodology uses a memory (i.e., case base) of experiences represented as cases, the next step is to understand how to select the experience, that is, case, and to reuse it properly. The question that needs to be answered is this, "What case in this memory has the most suitable solution I can reuse to solve my new problem"? The way it is answered in CBR is by relating the problem and the problems in the cases that make up the case base in such a way that the notion of "most suitable" is reflected. This relation was earlier referred to as *similarity*. The user will identify a problem in the base as very similar to the query problem if its solution is very useful.

Now suppose we have is a query problem and a case base to choose experiences from. There are many experiences in the case base but we do not know which one to

Table 2.4 Attributes in query problem, Case 1 and Case 2

Query problem	Attributes labels	Case 1	Case 2
Unusual car noise	Symptom	Unusual car noise	Unusual car noise
Knocking engine	Observations	Knocking engine	Knocking engine
3	Since last inspection (month)	4	14
No	Rhythmic pounding	Sometimes	No
No	Related to car speed	No	No
Yes	Oil pressure light flickering	No	Yes
No	Leaking oil	No	Rarely
What is to be determined	Solution	Loose transmission torque converter	Oil burning

take. The main difficulty we face is that the query problem may not be recorded in the case base because one cannot store all possible situations. Therefore, CBR has developed intelligent techniques to take advantage of the experiences even if they do not exactly match the query problem.

In order to illustrate how this is done we extend the example of car faults (Table 2.2) and look at a query problem and a small case base, containing just two cases, shown in Table 2.4. What we do is to compare the query problem with the problems of the stored cases. This comparison is a crucial step and known as *similarity assessment*. The goal is to find a case that helps in solving the problem. In other words, the case should be useful for this purpose. The reason is that a case is useful if its problem description is close to the query problem. Similarity is just a word for this. The goal is the cases are analogous in such a way that their solutions can be reciprocally reused.

Assessing similarity between two cases represented with attribute-value pairs entails two concepts.

- (1) Similarity between attributes.
- (2) Relative relevance of each attribute.

When we compare two such cases it is natural to compare them attribute by attribute. This is the best way particularly when cases are represented through attributes. For this a similarity notion between attributes is needed. Each attribute requires its own similarity function. As a general rule, a similarity value 1 is given when the values of two attributes are the same; and a similarity value of 0 is given when the values are not the same. Along these lines, for two values that are different from each other but that may be considered medially similar, one can use a value of 0.5. This can, of course, become much more complex. These issues are further discussed in Chaps. 6 and 7, both about similarity.

In the example, such a similarity function would define for attribute “Oil pressure light flickering” a value of 1 if both cases we have the same value for this attribute, and 0 otherwise. The attribute “Leaking oil” could have a similarity function return a value of 1 if in case both cases we have the same value for this attribute; a value

Table 2.5 Comparisons between query problem and cases

Attributes labels	Sim (Query problem, Case 1)	Sim (Query problem, Case 2)	Importance of attributes
Observations	1	1	8
Since last inspection (months)	0.9	0.2	2
Rhythmic pounding	0.6	1	7
Related to car speed	1	1	2
Oil pressure light flickering (binary)	0	1	8
Leaking oil	1	0.9	3

of 0 if one case has a value of Yes and the other has a value of No; a value of 0.9 if one case has a value of Rarely and the other case has a value of No; and a value of 0.1 otherwise. These numbers resulting from the similarity function denote the degree of similarity.

The second concept within similarity assessment is the relative relevance of each attribute. In practice, each attribute is not equally relevant, and this has to be represented in the similarity assessment. In addition, the problem of describing the importance of the attributes is denoted by a number too. Larger numbers denote a greater importance because more important attributes play a larger role. The comparison is described in Table 2.5 and will later be described in more detail.

The comparison uses two parameters based on the two concepts already introduced:

- (a) Similarities between the values of the attributes between the two cars. These similarities are called local similarities.
- (b) The importance of the attributes. This is expressed in terms of integers where larger means more important. The numbers denoting the importance are called weights.

Presently, we assume that both parameters are given. Later we will better explain their meaning and potential sources.

Next, we want to compute the overall similarities of the query problem to the two cases. A simple and plausible way to do this is by taking weighted sums of local similarities with the weights as coefficients. If sim denotes the intended similarity, we get:

$$\text{sim}(\text{act}, \text{Case1}) = \frac{1}{30} \cdot (1 \cdot 8 + 0.9 \cdot 2 + 0.6 \cdot 7 + 1 \cdot 2 + 0 \cdot 8 + 1 \cdot 3) = 0.633,$$

$$\text{sim}(\text{act}, \text{Case2}) = \frac{1}{30} \cdot (1 \cdot 8 + 0.2 \cdot 2 + 1 \cdot 7 + 1 \cdot 2 + 1 \cdot 8 + 0.9 \cdot 3) = 0.936.$$

Therefore, we choose to reuse the solution from Case 2, which is oil burning. The reason is that Case 2 is more similar to the problem than Case 1 and there-

fore, according to our motivation, more useful. This can be formulated as a general principle.

Definition 2.4 Let CB be a set of objects and p be an object; then some s of CB is a nearest neighbour to p if there is no object in the CB that has a higher similarity to p than s .

The principle is that no case is more useful than a nearest neighbour. The advantage of having degrees of similarity is that we can compare them and have a way to determine which experience is closer to the new problem. In particular, it allows us to say which recorded experience is the most similar one. This brings up the usefulness of the concept of nearest neighbours.

For an investigation of the cars example we had to define an adequate similarity measure. For our method, this looks as follows for objects with the attribute-value vectors of an arbitrary domain: $x = (x_1, \dots, x_n)$ and $y = (y_1, \dots, y_n)$:

Attributes : Attribute₁, Attribute₂, ..., Attribute_n.

Similarity between values of attributes (local similarities):

$$\text{sim}_1(x_1, y_1), \text{sim}_2(x_2, y_2), \dots, \text{sim}_n(x_n, y_n); \quad x_i, y_i \in \text{dom}(\text{Attribute}_i).$$

Overall (global) similarity:

$$\sum_{i=1}^{i=n} (\omega_i \text{sim}(x_i, y_i) \mid 1 \leq i \leq n).$$

In the example, we did not discuss the origin of two numbers:

- (1) The similarity functions for each attribute.
- (2) The weights that should reflect the importance of each attribute.

The (local) attribute similarities are easier to get because they deal only with the domain of a single attribute and are therefore easier to estimate adequately. The weights, however, are of global character because they relate attributes to each other and are therefore much more difficult to determine. Intuitively importance means that an important attribute has a large influence on the choice of which case is the nearest neighbour for a query.

The case base in the example is very small but sufficient for illustrating these main concepts. We see that none of the cases has exactly the same problem as our query problem but the provided solution is still useful. There are two aspects that are over-simplified in the example:

- (1) We have only two cases and the decision between them is easy. If we have hundreds or thousands of cases, more sophisticated techniques need to be used.
- (2) Although the reused case problem is close to the query problem, it is not exactly the same. In the example we were lucky because the old situation could be the

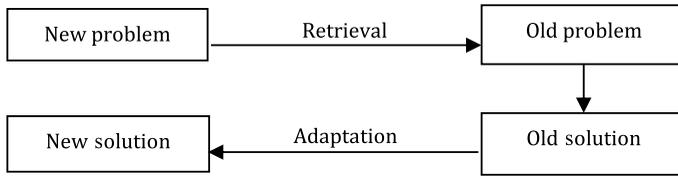


Fig. 2.3 Reuse principle

used unchanged. But suppose we had in the case a problem with front lights and in the actual situation exactly the same problem with back lights. Then it makes no sense to operate on the front lights. The advice is rather to do the same repair on the back.

The latter refers to an adaptation of the solution provided by the nearest neighbour. In general, *adaptation* takes place when one wants to reuse a solution with some modification.

2.8 Reuse and Adaptation

The use of cases is a reuse of previous experiences in a new situation. If the new problem situation is exactly like the previous one (which is supposed to have been successful) then the reuse is simple: Just copy the old solution. The general reuse principle for a selected case is shown in Fig. 2.3.

It is rare to be able to use a solution exactly as it is recorded. This happens if the new problem situation is not too different in essential aspects from the nearest neighbour selected from the case base. Then the recommendation is to adapt the recorded solution before reusing it to best suit the new problem. This can be done either manually or automatically. CBR presents formal adaptation methods, which we will introduce next.

Adaptation can be performed on different levels of granularity. One extreme case is reusing the solution strategy. An example is reusing a travel plan. Another extreme case is using the solution itself. Both are called solution adaptation.

Suppose we have to design exercise plans for people who need to increase their endurance. The simplest way would be to create a weekly plan for running. Now suppose there is a person who is not allowed to run because of knee problems. The previous plan can still be used but running has to be replaced by swimming or bicycling. The elements of the procedure are shown in Fig. 2.4.

In an abstract way we can describe the CBR problem-solving procedure by the following steps:

- (a) First describe the problem formally.
- (b) Search in the case base for the nearest neighbour and select it.
- (c) Make use of the retrieved solution by copying or adapting it appropriately.

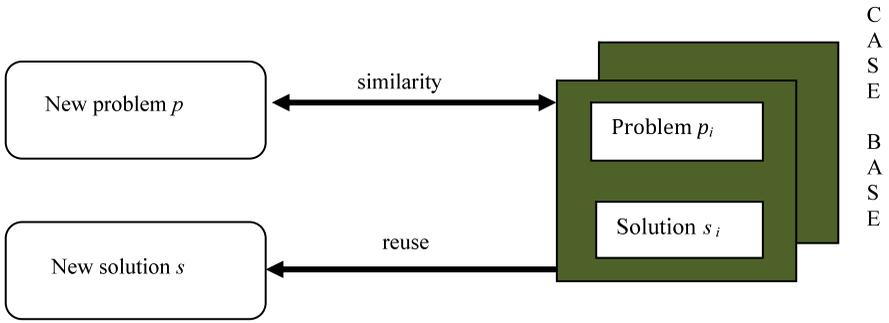


Fig. 2.4 Abstract CBR procedure

The approximate nature of case-based reasoning has the consequence that there is no guarantee that the chosen case provides a good solution. For instance, the case base may not even contain a good solution for the new problem. Sometimes this can be easily seen, as in symmetric problems. Take for instance an experience of a car problem with the solution “exchange the left bulb” when we have the same problem with the right bulb. It is not necessary to record this problem because we can simply adapt the presented solution. There are other situations where this is not so easy and a systematic evaluation is needed. This will be discussed in Chap. 9, Adaptation.

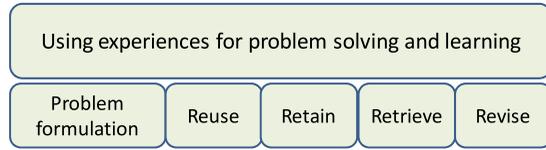
After adaptation, the adapted solution has to be tested in reality and possibly modified further. If the solution obtained in this way is satisfactory, then one may decide to add the case (new problem, final solution) to the case base in order to improve it. This last step can be interpreted as a learning step. More learning methods will be discussed in Chap. 10, Evaluation, Revision, and Learning. Adaptation allows case bases to be smaller than if no adaptation could be done. Furthermore, adaptation can be also extended by reusing a strategy when the solution is given because strategies can also be adapted. These methods are found in game playing. In chess, for instance, strategies are what is reused most often (but hard to formulate!)

If we solve a problem using experiences, there are many ways of doing it. For systematic reasons it is desirable to have a general process model for CBR problem solving. This will be discussed next.

2.9 Models of CBR

We now combine the understanding gained in the previous sections into two views on CBR. The first view considers the processes that take place when CBR is applied, that is, problem formulation, retrieve, reuse, revise, and retain. The second view considers the knowledge organisation in CBR. CBR systems store their knowledge in knowledge containers. The next two sections will describe these views in more detail.

Fig. 2.5 Tasks in the CBR process



2.9.1 CBR Process Model

Figure 2.5 introduces the first view we present of CBR, i.e., the main tasks the CBR methodology implements. In a more abstract way they will now be extended to a general process model that applies to the entire CBR methodology.

Figure 2.5 lays out the main tasks of this process. Although we have already described the main concepts of the process, in this section, we present these tasks in the context of the process model.

2.9.1.1 Problem Formulation

Problem formulation is a task that starts from the need to obtain a new problem from a user. Ideally, users should enter the new problem using the same representation and level of detail as those of the cases in the case base. Often, this is not the case. As an example, it may happen that the user knows what to achieve but cannot express a precise problem. Consider a user who wishes to find a “comfortable chair” for a living room. The problem formulation would need a description of chair parts and their properties that may not be available. Therefore, one cannot immediately describe such problems as cases.

This can be done in different ways. This is also known as the query generation problem. A somewhat oversimplified view is that the problem is stated exactly and complete with all details. In fact, it can be costly to acquire in an attribute-value representation the values of the attributes for the query problem.

An essential point therefore is to acquire as little information as possible for solving the query problem but enough to provide an answer. There are two major ways to proceed:

- (1) Use a specific, possibly standardized formulation of the problem.
- (2) Perform a dialogue with the user. This is discussed in Chap. 20, Conversational CBR.

After the content of the new problem is obtained, there are still different ways to formulate it physically. It can be typed into a computer; it can be spoken, or it can be represented as an image or diagram. These variations will be discussed in later chapters.

2.9.1.2 Retrieve

As previously mentioned, the goal of Retrieval is to determine the case that is most similar (i.e., most useful) to the new problem. Retrieval starts when the new problem is readily available and completes when a case is retrieved, becoming available for the next task of the process: reuse.

For purposes of simplification, we assume that only one case is retrieved. Variations are, of course, possible. They are further discussed in the chapters dedicated to retrieval, Chaps. 8 and 14.

Retrieval is comparable with a search, where the new problem is used for guidance and the case base is the search space. Retrieval is a demand and this demand has to be formulated. In order to formulate it one needs a set of search paths to select a successful one. The description of the paths is called an index structure. These indices are basic for the search. Depending on the search structure there are many indexing methods, for instance:

- For searching a book, the index is a page.
- Search for data entry uses a pointer to a record.
- Searching for a record in a database is a pointer to a record, realised by a key.

As previously mentioned, retrieval methods are not general; they have to be designed for each system. This is because of the complexity of cases and the inexact matching that CBR implements.

2.9.1.3 Reuse

Reuse is the step of the process when one case is selected for its solution to be reused. It is completed when the new solution is proposed for the next task of the process: revision. Reuse is about proposing a solution for solving the new problem by reusing information and knowledge in the retrieved case(s).

Reuse is quite simple when the new problem is identical to the retrieved case problem. When they differ, they require adaptation. This is a general theme; details are in Chap. 9, Adaptation.

2.9.1.4 Revise

Revise starts when a solution is proposed to solve the new problem, and it is completed when it is confirmed. Revise aims to evaluate the applicability of the proposed solution. Evaluations can be done in the real world or in a simulation. Simulation is easier and cheaper but may neglect practically important aspects. In the real world, evaluation aspects may be present that one might not have considered in the model. In fact, this is an old phenomenon in Artificial Intelligence called the frame problem. It says that one can never completely formulate all possible facts that may occur in the real world.

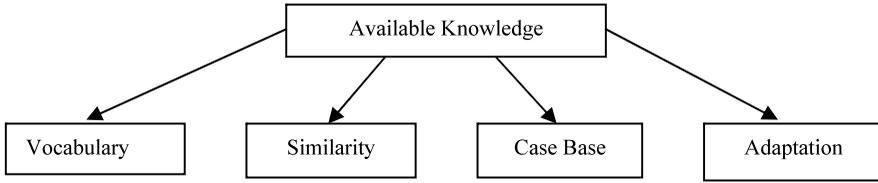


Fig. 2.6 Knowledge in CBR

2.9.1.5 Retain

When revising generates a new case, updating the case base with the new (learned) case for future problem solving takes place. Nevertheless, a confirmed solution may or may not be retained. Some systems learn new solutions adapted through use; others accept only actual cases. Revise and retain are discussed in Chap. 10, Evaluation, Revision, and Learning.

This model is detailed and extended in various ways. The usefulness of having a process is that such improvements can smoothly be integrated. For instance, the learning aspect is much more complex; in the cycle, cases can only be added but not forgotten. This is connected with the maintenance issue that is discussed in Chap. 11, Development and Maintenance, together with the problem of developing a CBR system.

This view of CBR lists the main tasks the methodology entails. Another perspective on CBR is given by the knowledge containers it requires to be successfully implemented, which is discussed next.

2.9.2 CBR Knowledge Model

The knowledge container view of the CBR methodology is based on the perspective that CBR is a knowledge-based system. Knowledge-based systems are a class of intelligent systems that are designed by having a knowledge base in an independent module. In CBR, we extend this notion to emphasize how the methodology utilizes different kinds of knowledge in distinct repositories: the knowledge containers. While the tasks listed in the previous Sect. 2.9.1 look at CBR from the process point of view, one may also ask what kind of knowledge is represented and where it can be found. Knowledge can either be represented explicitly or be hidden in an algorithm. In any case, there must be some way to formulate the knowledge; we say that knowledge is presented in some formulation. The formulation is stored in what is called a knowledge container.

For the knowledge containers described next we state what kind of knowledge could be contained in them. We say little about how the knowledge is formally represented. In CBR we identify four major knowledge containers. They are presented in Fig. 2.6.

The knowledge containers represent one view of a CBR system; they are not modules that can perform certain subtasks. They contain certain knowledge units that in combination help solve a problem. Next, we give a short overview of the containers that will be extended in the following chapters.

2.9.2.1 The Vocabulary Container

The vocabulary is basic for any knowledge-based system. This is not special to CBR. The vocabulary determines what one can discuss explicitly.

The vocabulary plays a role in all levels of abstraction, which is illustrated by very simple examples:

1. If we do not know the word heart rate we cannot talk about it. It is knowledge that this term plays a role.
2. If the term tax cost is missing one cannot compute the tax correctly. Again, this is knowledge. This aspect plays a major role in different countries, where different tax regulations are involved.

The vocabulary container retains knowledge about how to explicitly describe the knowledge elements being used. This does not depend on the types of descriptions, ranging from logical constructs to free text. It is a classical observation in science that the solutions of difficult problems have been found only after some person introduced a new crucial notion.

Therefore, there is usually much knowledge contained in the chosen vocabulary. For a real-world object there are in principle infinitely many terms that have something to do with the object but only a few are relevant for a specific task. That means an object can (and should) have different description terms for different tasks.

In the vocabulary container one can identify various sub-containers that are useful for technical purposes as retrieval, input or output. These are, for example, names of employees, companies, products in a supermarket, and so on. These sub-containers are frequently defined and used in application domains.

2.9.2.2 The Similarity Container

The knowledge in the similarity container consists of all knowledge needed to determine what makes a case similar to another such that their solutions can be reciprocally reused. There are multiple ways to ensure similarity knowledge accomplishes this: From the use of simple symbolic similarities where the values are either equal or not, through the use of weights to represent relative importance of the attributes, through the use of systems where relevance is computed at runtime, to the use of fuzzy algorithms that consider all attributes and their importance at once.

The similarity is used for retrieval purposes. This means that something has to be known about the problem and what is required for the solution. As an example we

consider the task of squaring numbers and assume we are unable to multiply and do not want to learn how to do so. Suppose we have a base of solved problems, say

$$Squ = \{(2, 4), (2.5, 6.25), (-3, 9), (-5, 25), \dots\}.$$

As a special problem we take square $(3) = ?$ The answer is not in our list; therefore we have to look for the nearest neighbour of “3”. A first try is to take the Euclidean distance, which gives 2.5, and the answer 6.25. A much better method is to equip the similarity measure with the knowledge square $(x) = \text{square}(-x)$ for all x . Then we would retrieve -3 which gives the correct answer. The similarity measure is much easier to use than it is to learn multiplication. In Chaps. 6 and 7, similarity concepts are studied in detail.

For CBR and retrieval purposes it is important to quantify similarities. This is done by similarity measures, which can be defined as a mapping

$$\text{sim} : U \times U \rightarrow [0, 1]$$

where U contains the objects to be compared.

Not all aspects of a problem situation may be of equal importance. For example, the price of a car may be more important than the colour. If the similarity knows this then it would pay more attention to the price attribute than to the colour attribute. A way to make this possible is to assign weights to attributes. Earlier, we saw an example dealing with car repairs where the similarity measure was naively chosen but successful. It ranked the cases and we selected the most similar one because similarity tends to be an adequate proxy for utility.

2.9.2.3 The Case Base Container

The case base container contains experiences as cases. These experiences may be available from the past or may be constructed from variations of existing cases, or be completely artificial. The description of the case base as a knowledge container is straightforward as the case base is typically the main source of knowledge in CBR systems. The implications of the case base as a container of knowledge are discussed in multiple chapters. Representation formalisms are discussed in Chap. 5, Case Representation; quality and maintenance are discussed in Chap. 11, Development and Maintenance.

2.9.2.4 The Adaptation Container

The knowledge in the adaptation container will be used to adapt cases to solve new problems. The most common formalisms adopted for adaptation are rule bases; nevertheless, case bases can be used, and even existing cases from the case base have been used at runtime to extract adaptation knowledge. As previously described in Sect. 2.9.2.4, the knowledge in the adaptation container can be used to transform

an existing solution or generate a new solution based on a strategy from a previous solution.

In the adaptation container one finds information on how to modify a solution. In the adaptation container rules are stored for adapting a retrieved solution to a new situation. Such rules are intended to perform a solution transformation that has to take care of the fact that the solutions obtained from the case base using the nearest neighbour principle may still be insufficient (either because of a not very well defined similarity measure or simply because the case base does not contain a better solution). In this situation the solution is adapted. Adaptation knowledge can drastically reduce the number of cases needed in the case base. More is shown in Chap. 9, Adaptation.

2.10 Tools

Tools can speed up design and assessment of an application. We list some tools that are currently available. However, given the dynamics of tools, we recommend that the reader rely on a more agile source, like the Cbrwiki (2011).

In addition, we mention some general-purpose tools, i.e., tools that can be used for building a general CBR system and using it for many applications. Using such systems one can avoid a lot of work, not least because of a graphical user interface with useful visualisation.

Some major examples follow.

- (1) CBRWorks (<http://cbr-works.net>) and Orange (Schumacher 2002). CBRWorks is developed for e-commerce applications but can be used for other purposes also. It contains elements from all knowledge containers and can perform the full CBR cycle. Orange is a further development and has a more powerful retrieval engine.
- (2) myCBR. It is open source, developed under the GPL license. It can be viewed as a successor of CBR Works and contains many useful features. See myCBR (<http://mycbr-project.net>), from where it can be downloaded; this also contains a tutorial.
- (3) jColibri. It is a general framework that supports many features like graphical interfaces, description logics and ontologies, textual CBR, evaluation, and so on (<http://gaia.fdi.ucm.es/projects/jcolibri/>). jColibri 2 (Recio-García et al. 2013) has added a number of features and is becoming more and more a reference tool for teaching and research purposes.
- (4) CBR in Microsoft[®] Excel. For users familiar with macros in Excel, a simple case retrieval system can be developed in it. A worksheet should be reserved for cases, with their attributes laid out vertically. A different worksheet is used for retrieval, where the new problem is compared to all cases in the case base. Note that the retrieval worksheet will require one column of computation of similarity for each case. Weights can be listed in a separate sheet and called from the retrieval worksheet. Solutions can be presented in a separate sheet. Such implementation can be extended to include a validation method.

2.11 Chapter Summary

The chapter presents the basic notions used in CBR and necessary for understanding the remainder of this book.

Case-based reasoning is a reasoning methodology for problem solving. It mainly relies on experiences in which problems were solved in the past. CBR reuses previous experiences to solve current, new problems. Problem solving experiences include problems and solutions. Problems and solutions should be explicitly stated in order for the experiences to be successfully reused. CBR can be used to perform multiple reasoning tasks, such as classification, planning, and design. The way to develop a reliable CBR system is by limiting its scope to one single reasoning task. Such a system would be populated by cases that describe experiences of performing the single chosen reasoning task in a given target domain.

The simplest method to represent cases is to use attribute-value representations. With a limited and previously defined set of attributes, each case is populated with individual values for each attribute. This representation allows a case comparison at the level of attributes.

Cases are compared to search for a similar case. Problems are submitted to a CBR system through what we call query problems. Once a new query problem is formulated through the set of attributes defined for case representation, similar cases can be retrieved.

Case retrieval utilizes a similarity measure to search for similar cases whose solutions may be reused to solve the new query problem. How to assess similarity between cases is a core method in CBR.

The problem in the retrieved case is typically very similar to, but not exactly the same as the query problem. This may cause the solution in the retrieved case not to be perfectly suitable for solving the new query problem. Adaptation is the step that modifies the solution in the retrieved case in order to make it perfectly suitable for solving the query problem.

There are two models of CBR. The CBR process model incorporates formulating the problem, retrieving solutions, reusing them, revising and repairing them, and storing them as new experiences.

The CBR knowledge model describes the containers where knowledge is stored. There are four knowledge containers: Vocabulary, Case Base, Similarity, and Adaptation.

From reading this chapter, the reader has a deeper understanding of the CBR process. However, we recommend you do not yet jump into designing your own CBR system, not until after reading Chaps. 3 and 4; the technical details are presented in Part II.

2.12 Background Information

The first substantial publication on case-based reasoning is the 1993 book by Kolodner (1993). It introduces the main problem areas, thoroughly describing case representation, structure, indexing, retrieval, adaptation and learning.

CBR has roots outside of computer science, mainly in cognitive science, psychology, and language understanding. The first CBR systems were built within this context. The use of analogy for reusing previous events is discussed in Carbonell (1983). The many roots of CBR today are discussed in Richter and Aamodt (2005).

One of the most cited foundational articles is the 1994 article by Aamodt and Plaza (1994). The CBR cycle as presented by Aamodt and Plaza is a simple and complete way of visualising the CBR methodology as a whole. It introduces the CBR cycle and names the four R's in the cycle: *retrieve*, *reuse*, *revise*, and *retain*. An early cycle for modelling the CBR process, referred to as a CBR flowchart, is given in Riesbeck and Schank (1989). The CBR cycle was extended in many ways to describe additional activities like maintenance and learning. Some of these extensions are described by Bridge (2005). More historical information is in Chap. 1, Introduction, and in Chap. 23, Relations and Comparisons with Other Techniques.

The knowledge containers were introduced by Michael M. Richter; see, for instance, Richter (1998). We return to them in Chaps. 10 and 11 on learning and on development and maintenance. When systems are developed or improved, the contents of the containers are the objects of interest.

The example in Fig. 2.1 is based on a problem discussed on Car Talk from National Public Radio® on 19 February 2011.

2.13 Exercises

Exercise 1 Suppose you are in the automotive domain. Look at the three contexts manufacturing cars, marketing cars and repairing cars. Find for each context typical attributes that would not be used in the other contexts.

Exercise 2 Describe the purpose of shifting knowledge from the case base to

- (a) the similarity measure,
- (b) the adaptation container.

What is the influence on the size of the case base?

Exercise 3 Give an example where the retain step of the process model does not improve the performance of the CBR system.

Exercise 4 (Intended for readers who understand databases) Write a process cycle for databases. Can you identify some knowledge containers?

Exercise 5 (Intended for computer scientists) Name some knowledge containers for other knowledge-based systems such as rule-based reasoning, fuzzy expert systems, and ontologies.

Exercise 6 Find useful sub-containers for adaptation.

Exercise 7 Propose an application domain where CBR can be used to provide solutions to problems. Consider what source of cases you would have.

Exercise 8 Describe characteristics that you would require for a problem to be solved with the CBR methodology.

Exercise 9 Describe an area of expertise that you master, e.g., playing a game. Describe how you would explain to someone what makes cases similar to others so that their solutions can be swapped with minimal adaptation.

Exercise 10 Elicit similarity knowledge from an expert (not you) in any domain in which you are not a master. In other words, elicit for the expert's domain of expertise what makes cases similar to others so that their solutions can be swapped with minimal adaptation.

Exercise 11 Name one AI methodology and an example problem you are familiar with and then list advantages and disadvantages you see when comparing it with CBR.

References

- Aamodt A, Plaza E (1994) Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Commun* 7(1):39–59
- Bridge DG (2005) The virtue of reward: performance, reinforcement and discovery in case-based reasoning. In: *Keynote at ICCBR 2005: 6th international conference on case-based reasoning*. Chicago, IL, USA, August 2005. Awards, Honours, Affiliations. <http://www.cs.ucc.ie/~dgb/recognition.html>. Accessed 28 Feb 2013
- Carbonell JG (1983) Learning by analogy: formulating and generalizing plans from past experience. In: Michalski R, Carbonell JG, Mitchell T (eds) *Machine learning: an artificial intelligence approach*. Springer, Berlin, pp 137–159
- Cbrwiki (2011) Case-based reasoning Wiki. <http://cbrwiki.fdi.ucm.es/wiki/index.php>. Accessed 18 Jul 2011
- Kolodner JL (1993) *Case-based reasoning*. Morgan Kaufmann, San Mateo
- Recio-García JA, González-Calero PA, Díaz-Agudo B (2013) jColibri2: a framework for building case-based reasoning systems. *Science of Computer Programming* (in press)
- Richter MM (1998) Introduction. In: Lenz M, Bartsch-Spörl B, Burkhard H-D et al (eds) *Case-based reasoning technology: from foundations to applications*. Lecture notes in artificial intelligence, vol 1400. Springer, Berlin, p 1
- Richter MM, Aamodt A (2005) Case-based reasoning foundations. *Knowl Eng Rev* 20(3):1–4
- Riesbeck CK, Schank RC (1989) *Inside case-based reasoning*. Erlbaum, Hillsdale
- Schumacher J (2002) Empolis Orange—an open platform for knowledge management applications. In: Minor M, Staab S (eds) *Experience management: sharing experiences about sharing the experience*. Papers from the 1st German workshop on experience management, Berlin, March 7–8, 2002. GI, Bonn, p 61

Chapter 3

Extended View

3.1 About This Chapter

This chapter extends the view and consequently the use of CBR systems as described in Chap. 2, Basic CBR Elements. It requires thus basic understanding of Chap. 2. The main new aspect is that one does not compare anymore old problem situations with new ones but rather compares problem situations with possible solutions. An important point is that all the techniques developed so far can be used almost unchanged.

3.2 General Aspects

In this chapter the basic concepts of CBR as they are given in Chap. 2 are now extended to situations where experiences vary in many ways. In Chap. 2, Basic CBR Elements, we discussed the CBR methodology with respect to experiences that were problem-solution pairs; let us now refer to this as the standard view. We now focus on certain types of experiences that do not conform to the standard method. This extended view considers the relation of usefulness between problem situations and possible solutions as shown in Table 3.1. It implies that one has to analyse the candidate solutions with respect to their usefulness for solving the problem.

The main new aspect is that now similarity measures have to compare problem situations with possible solutions. An important point is that all the techniques developed so far can be used almost unchanged. The major applications of the extended view are in e-commerce. By adding this extended view, we can cover more problems and deal with more situations while still benefiting from methods developed for standard CBR.

Experiences are fundamental to almost all of human reasoning. The development of cultures depends to a major degree on what has occurred in the past. However, the experiences are encountered in different forms, and are recorded and used in different ways. In Chap. 2, Basic CBR Elements, we presented and discussed how to use experiences from the CBR standard view. We described different methods of recording and using experiences.

Table 3.1 Standard and extended CBR views

	Standard CBR view	Extended CBR view
Experiences	previous problem-previous solutions	previous solutions
Similarity between	query problem-old problems	query problem-possible solutions

Technically, we described the CBR methodology in Chap. 2, Basic CBR Elements, in two ways. By the process and by the knowledge models. We emphasized that CBR relied heavily on using experiences stored in a case base. We also discussed that other knowledge was stored in the remaining containers. This leads us to the question: “What is special about CBR”? In our view, the essence of CBR is presented through its models in Chap. 2, Basic CBR Elements. One is the process model, or CBR cycle, which shows how its steps make use of the knowledge containers (i.e., the second model). In this chapter, we investigate how to use the characteristics and the techniques of CBR even when the experiences in the case base are no longer explicitly recorded as problem-solution pairs.

Does this mean the knowledge contained in the cases is no longer needed? Not at all; but it has to be compiled into containers, mostly into the similarity container. An advantage is that one can handle situations where such experiences are not available in the previous form or may not even exist; in addition, they may not be necessary.

Next, we list some situations in which this extension is applicable:

- (a) E-commerce. One typically records sales but the records do not necessarily include a description of the buyer’s demands in the past, because one does not know for what the shoppers have been searching. This knowledge is partially available and not totally forgotten; it is, rather, compiled into the similarity measure between queries and solutions.
- (b) Information retrieval. A user has a problem and the solution is in a document. When a user searches for a document in order to solve a problem neither the original query nor the documents retrieved or selected are recorded.
- (c) Recommender systems. Audience statistics from TV and other media are available but they do not include individual information about the users.
- (d) Question answering systems abound on the Internet and embody valuable answers to questions but no context.

The following sections discuss the extended view further.

3.2.1 E-Commerce

We start with a very simple but motivating example in car sales. This will illustrate how to apply CBR without referring to explicitly recorded sales events.

A used cars dealer has cars to offer and there is a prospective buyer. The buyer has certain preferences and wishes to satisfy them as much as possible. The dealer has many cars in stock and describes them using features. Some of them may match to some degree the buyer’s preferences. The cars are stored in a product base.

Table 3.2 Comparison between demand and Offers 1 and 2

Demand	Attributes	Attribute importance	Offer 1	Sim (<i>dem, Off1</i>)	Offer 2	Sim (<i>dem, Off2</i>)
Coupe	Body	8	Coupe	1	Coupe	1
15,000	Price	8	16,500	0.9	0.6	19,000
4	Doors	8	4	1	1	4
60,000	Miles	5	45,000	1	65,000	0.8
150 mph	Speed	2	140 mph	0.9	160 mph	1
Black	Ext. Colour	2	White	0	Dark grey	0.8
2004	Year	3	2003	0.9	2000	0.6
?	Desired/Offered Car		Ford XY		Mustang Xi	

The car dealer now wishes to select a car that matches as much as possible the buyer’s demand in order to realise a sale. It is not likely that the demand is totally satisfied. Here, the similarity enters the stage. Both the demand (the buyer’s problem) and the offered car (the dealer’s solution) are descriptions of cars expressed as attribute-value vectors.

If one would have sales recorded as experiences in the standard way, one could compare old demands to new demands. But as long as we know (i.e., have the knowledge of) how a feature of an available product (possible solution) can meet a specification of a demand, we can compare the specification of a demanded car to available cars. It is likely that both methods would take us to the same solution, although the latter is more direct. It is more direct in that it would need less adaptation. Furthermore, we have no records of previous demands, so the latter is the only choice.

3.2.1.1 Representation

We describe the problem in the same way as in previous example. Table 3.2 shows an example of how one can compare the buyer’s demand with Offer 1 and Offer 2.

We compute the similarities as we did in Chap. 2, Basic CBR Elements. As a result, the customer would find Offer 1 to have higher utility given it is more similar, as demonstrated by the higher similarity score:

$$\text{sim}(dem, Off1) = \frac{1}{36} \cdot (1 \cdot 8 + 0.9 \cdot 8 + 1 \cdot 8 + 1 \cdot 5 + 0.9 \cdot 2 + 0 \cdot 2 + 0.9 \cdot 3) = 0.9,$$

$$\text{sim}(dem, Off2) = 1 \cdot 8 + 0.6 \cdot 8 + 1 \cdot 8 + .8 \cdot 5 + 1 \cdot 2 + 0.8 \cdot 2 + 0.6 \cdot 3) = 0.8.$$

In this example, as it often is in sales, there is no structural difference between demands and products. In fact, demands are nothing more than possible products that, however, may not exist. It is irrelevant to the similarity measure which car is demanded or which one is offered. Hence the similarity search for a nearest neighbour in the product base is expected to provide the product that best meets the demand,

that is, the best available car for the given demand. This product is the suggested solution to the car buying problem. Now consider what we can learn from this example. First, we have an observation: The similarity measure takes into account the importance of the attributes that are expressed in terms of weights.

The weights embed experiences about the preferences from data of preferences from past sales. These are compiled into the measure. They reflect what customers preferred in the past.

Now observe that the suggested car is not perfect for the demand, which was a black car while the offered one is white. Such a change is an adaptation. One problem is that not every physical possible adaptation can be performed. For instance, there is no legal action to change the year of the car. This is again the same as in the standard CBR view. This will be discussed in detail in Chap. 9, Adaptation.

The success in the car example was due to the fact that the seller could use the same attributes that the buyer used. But suppose the cars are sold, to buyers with mainly unknown interests, say in a different country with different laws, culture and economy. Then the seller would have a serious problem.

Buyers are interested in maximal utility when they buy a product. The utility refers to perceived advantages delivered by the product. Therefore, the utility refers to the intended functionality. This frequently occurs in a query or demand in e-commerce, although it applies much more generally to problem solving. The difficulty now is that the product may not be easily described because the intended use cannot always be foreseen. Moreover, functionality and product properties are mostly described in completely different terminologies. This results in a serious gap between problem and solution vocabulary.

An example is the association between a machine and its intended functionality. The producer describes a machine from the technical point of view, for instance, how many megabytes can be stored on a disc. The buyer on the other hand wants to know, “Can I store all my images here”? An example from e-commerce will illustrate the problem. Suppose we want to deal with electronic switches, we have a description of the intended functionality and we have a description of the technical details of the products, as in Fig. 3.1.

The example in Fig. 3.1 shows how serious the gap can be. It is kind of obvious that producer and customer will have problems understanding each other. On the left we have the buyer’s terminology with regard to the desired functionality and on the right we see the seller’s terminology with regard to the properties of the manufactured product. The gap is not only a purely linguistic one that could be resolved by a simple language transformation. The deeper problem is semantic. Both sides address different aspects. A skilled salesperson would be able to interpret the buyer’s demand in terms of product properties but this is difficult to automate due to the need for deep understanding. For this reason a mediator is needed who knows the language and intentions of both parties.

Definition 3.1 The gap problem is how to build a bridge between the problem and the solution descriptions. The bridge attributes are those that occur on both sides; in the example there is only one (voltage) and it is not very informative. If all attributes are bridge attributes, one can perform the comparison directly. In the situation such

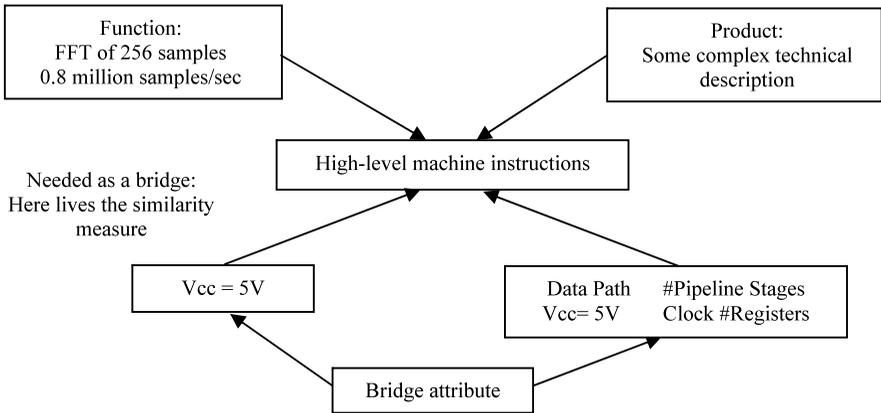


Fig. 3.1 Complex technical description

as the one here (only one bridge attribute), one has to introduce a whole new bridge level. This is a language level that can be reached from both sides and is understood by both parties, that is, a language to which expressions from both sides can be mapped. In this case, this is a high-level machine instruction language to which both, functions and products, can be mapped. It is also a problem in database design, where it is called a mediator level.

When one formulates the demand as an intended functionality, the product has to realise the functionality. This relation can be quite complex due to the potential complexity of product descriptions. For example, consider product descriptions of aircraft, armoured vehicles, and software.

The experiences in the extended view may bring up the question of why we would need two different concepts, namely, similarity and utility, if they have to correspond to each other. In e-commerce, the utility is in the functionality of the product. The similarity is in the product description. These two concepts are related to different aspects, the functionality and the product description. In practice, only the similarity is used for retrieval purposes, because it is the similarity that is used to compare functionalities and product descriptions while the utility is not used directly in the process.

As indicated, it is typical that one product is described from different views and contexts, for example:

- Customer view.
- Producer/manufacturer view.
- Marketing view.
- Tax view.

Sometimes one needs to compare different views, for example, when a company offers products and repair services. In addition to having multiple aspects and perspectives, objects may also have many descriptions. The elements of the solution

base (the *case base*) can be represented in very different ways. Besides the representations mentioned for cases through sets of features or texts, there are others:

- (a) Images. They occur regularly in medicine, but also in aerial views, traffic views, and so on. Images are treated in Chap. 18.
- (b) Audio. Music can also be a product. As a product, its easily identified features may not suffice for determining utility.
- (c) Diagrams. They are often used to describe characteristics for functionalities that may not be obvious.

We will discuss these aspects in Part IV. All these representations ask for special and often very sophisticated similarity measures. In all these interpretations similarity is no longer coherent its use in everyday language. We still use the term similarity not only for historical reasons but also because of the fact that all CBR techniques developed before apply here as well. In particular, all considerations concerning the process model and the knowledge containers remain valid. Also, the results, the retrieval mechanisms, and the rule methods remain unchanged.

In most of these examples the gap problem is drastically visible. One sees a relation between problems and solutions but a similarity measure is not easily available. One of the reasons is not only that the description languages for problems and solutions may be very different but also that there is a semantic gap. The next section discusses the use of the extended view in recommender systems.

3.2.2 *Recommender Systems*

Recommender systems are closely related to the sales situation. Instead of selling products one can also recommend them. The difference between an offer and a recommendation is as follows:

- (a) An offer is oriented to a demand only. If the demand is not precise enough, a dialogue tries to gather more details on it.
- (b) A recommendation does not need to answer a specific demand. In addition, a recommendation takes the user preferences into account, when available.

The task of a recommendation is to mention products to customers that are likely to satisfy their needs, desires, or preferences. For this, a simple list of products is insufficient. What is needed rather is knowledge on both, products and customers. In principle, this is also some kind of experience, although not in the standard form. Mostly, experiences used in recommender systems are recorded in the form of statistics.

This means recommenders ultimately address the information overload given a multitude of products. They use customer information and knowledge as a filter to select the products that are more likely to satisfy customers.

There are two main types of recommender systems: The ones that use content and the ones that use data-intensive methods without explicit knowledge. Case-based recommenders use content and this is the category of recommenders we discuss here.

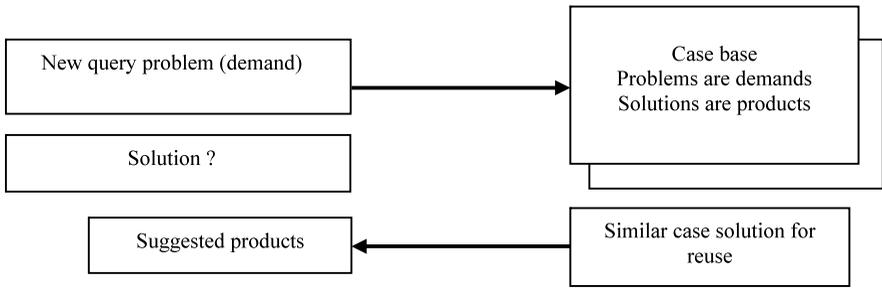
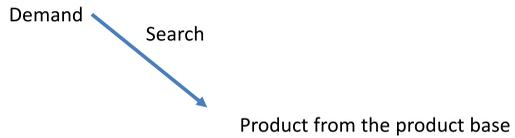


Fig. 3.2 Standard CBR

Fig. 3.3 Solution search in the extended view



3.2.2.1 Case-Based Recommender Systems

In principle, recommendations need also a query. This query is, however, not asking for a specific product. It is, rather, in the form of a more general wish. The wish may not even be formulated explicitly by the customer. If there is a user profile, one may generate or construct a wish that the customer is likely to have.

The answer or solution to the query problem is an item or a set of items that the customer may want. This requires knowledge about customer’s preferences. One way to obtain such knowledge is by recording the history of customer transactions. However, they are not always available on a personal basis.

Due to the nature of recommendations, it is common that many cases are retrieved and offered as recommendations to a user. In these situations, an ideal set of recommended items should not only match the buyer’s profile but also be diverse. Diversity methods are discussed in Chap. 14, Advanced Retrieval.

3.3 Extended Model

The extended view will now be discussed further. We have a demand, which we still call a query problem. In the context of sales, the solution is called a product. If we follow standard CBR using experiences, we would compare the demand with previous demands, as in Fig. 3.2.

The pair (demand, product) is a case. In the extended view, we need to realise the following, as shown in Fig. 3.3, as an extension of the standard approach.

We would like to organise the search again as a nearest neighbour search. This requires a notion of similarity between the problem and the solution (or the demand and the product). For the customer this is invisible and does not change the way she

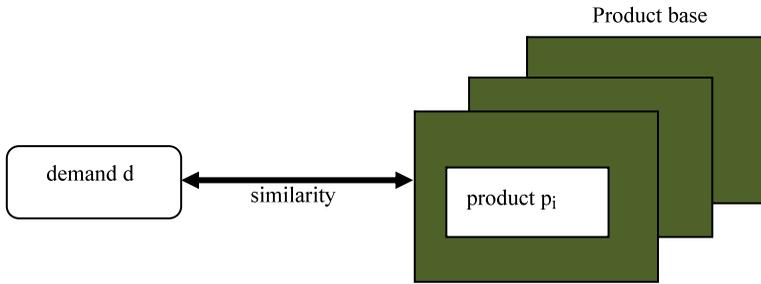


Fig. 3.4 The extended view

Fig. 3.5 The new sub-containers: query container and solution container



or he has to behave, regardless of whether the standard or the extended method is used. The customers describe their intended wishes and obtain an answer. They do not have to know how the answer is obtained, by experience or in some other way.

The choice of the extended method is only a little different for the sellers and the system developers. The sellers describe properties of their products. Instead of a case base there is now a product base that contains the solutions to the demands. The leading and unchanged motivation in such situations is still the view that similarity approximates or imitates utility. In particular, and most importantly, CBR is not restricted to the use of previous experiences, as illustrated in Fig. 3.4, which is completely analogous to the standard CBR view.

The next step is to generalize the approach further in such a way that the relations between the data structures of demands and products are even less obvious. This requires several changes to the standard approach. The first change affects the case base. In the standard view of CBR, the case base contains the solutions as the second component of the cases.

We replace the problem space by a general query space Q and a solution space that is denoted by S . The next step is to replace the case base by the solution base S . As a consequence, the case base container is also replaced by two knowledge sub-containers, the query container and the solution container, as seen in Fig. 3.5.

The query set is, in principle, easy to visualise. It contains all queries that the syntax allows.

Because of their frequent use in e-commerce, the solutions are often called products, which are elements of a product base. This is the main source of recorded experiences. Often we still use the word “case” for the solutions, and the solution container is usually called the case base.

Obviously the domains for similarity measures have to be generalized because objects of different kinds have to be compared. This allows two arguments of the measure to come from two different sets.

Definition 3.2 A general similarity measure is a function

$$\text{sim } Q \times S \rightarrow [0, 1].$$

This measure should approximate the usefulness of the solution for the query. It is apparent that one has to consider problem situations rather than just problems for adequate modelling. Similarity measures are discussed in detail in Chaps. 6, 7, and 13 on similarity.

3.4 More Generalizations

When one looks at the general sales situation, the question arises as to where the CBR techniques may also be useful. The goal is to deal with such situations by applying the techniques without losing the advantages described so far. This obviously needs a conceptual extension of the previous approach. Now we will further explore the extended view in order to get a feeling for its capacity. Most CBR techniques are still applicable. As mentioned, the changes are invisible to the user; they affect the internal system structure only.

There are many techniques for solving such problems and CBR is only one of them. In Chap. 23, Relations and Comparisons with Other Techniques, we will discuss some of them. The point is not to view this as a competition. We are interested rather in problem types where CBR may be helpful.

In the sales example, again an attribute-value representation was used. As we observed, the choice of the vocabulary was made with respect to the intended goal, namely, selling and buying the object. In the diagnosis situation from Chap. 2 this vocabulary would make little sense.

From the view of CBR techniques we encounter the same situation as before:

- We have a problem. This could be about searching for a desired car or a very general object.
- As solutions we may have very complex objects where it is not immediately clear how useful they are.
- The similarity measure selects the most useful solution to the problem.

Next, we will present a number of examples in order to indicate the wide scope of possibilities. Special techniques have been developed for some of these possibilities. Several are briefly discussed in Chap. 23, Relations and Comparisons with Other Techniques. A short overview is given in Table 3.3.

Further generalized views consider measures as *partnership measures*. The purpose of partnership measures is that both objects cooperate more or less well as good partners. This has a long commercial tradition, and has been fairly refined by marriage institutes that are in the so-called matchmaking market. In these services, each candidate gets a questionnaire that has to be filled out. These queries constitute the attributes and the customers give the values.

Table 3.3 General problem-solution examples

Problems	Solutions
Desired products	Available products
Symptoms	Therapy
Images	Meaning
Texts	Interpretation
Knowledge needed	Documents
Functionalities	Machines
Workers	Co-workers, partners
Questions	Answers

A more radical change came when similarity measures were regarded as a form of *dependency measure*. Dependency introduces some kind of partial ordering and $\text{sim}(x, y)$ expresses the degree of dependency of y from x . This includes the fact that one object may support another object to some degree. This direction is followed when a company is searching for employees.

This shows the great variety of generalizations. Some have been commercialized, others not.

In Table 3.4 we summarize these observations and comments and show the commonalities and differences between the standard and the extended view.

This means that the basic methods will work in the extensions. The concept of the process model and the knowledge containers remain valid. Tools take advantage of this by not reinventing the methods again.

3.5 Tools

The jColibri tool (<http://gaia.fdi.ucm.es/projects/jcolibri/>) offers particular support for recommender systems. CBRWorks is a commercial product (<http://cbr-works.net>). CBRWorks is developed for e-commerce applications but can be used for other purposes too. This tool is of interest to lecturers who want to do exercises with their students. One can easily perform interesting applications with moderate effort. It contains elements from all knowledge containers and can perform the full CBR cycle.

Table 3.4 Comparison between standard and extended views

	Vocabulary	Query	Answer	Similarity measure	Base	Adaptation
Standard view	Problem dependent	Problem	Solution	Problem-dependent	Cases	By rules, actions
Extended view	Same	Demand	Product	More general domains	Product base	Same

3.6 Chapter Summary

The chapter extends the view of CBR as presented in Chap. 2, Basic CBR Elements, which adopts experiences of the type problem-solution. That standard view is now extended to experiences that can be of different natures.

There is still a query problem and a solution, and a similarity measure. The similarity attempts to identify the relation of usefulness between problem situations and possible solutions. The similarity may need to compare two objects with different data structures and the measure has to respect this. Query problems are now explicitly located in a query set Q and solutions are located in a solution set S . These two sets replace the case base container. An important point is that the standard CBR techniques and methods can still be used with almost no changes.

Situations where this extension is applicable occur, for example, in e-commerce and in recommender systems. In e-commerce, query problems are demands that are to be realised by offers. The similarity measure matches demand specifications with product features to select the product to offer.

In recommender systems, solutions are oriented to profiles of customers. No specific demand is available, but it is still possible to make useful recommendations. Case-based recommender systems can make wide use of experiences that associate items in transactions that co-occur.

3.7 Background Information

The extended view developed slowly. It originates from challenges in e-commerce and recommender systems that motivated the generalization to similarity reasoning without experience. Early papers of interest are Wilke et al. (1998), Bergmann et al. (2001), and Smyth and McClave (2001), which included the consideration of diversity discussed in Chap. 14, Advanced Retrieval. A comprehensive overview of CBR and e-commerce is given in Bergmann (2002); of recommender systems in Burke (2000). The approach for recommender systems shown here follows O’Sullivan et al. (2002), but see also Montaner et al. (2002). There are two main approaches to recommender systems. The one that uses CBR is widely known as content-based where memory-based systems represent characteristics of products. The other approach employs statistics and machine learning. An overview is given in Su and Khoshgoftaar (2009).

3.8 Exercises

Exercise 1 The task is the development of an Internet shop for selling used cars. Cars are represented by attribute-value descriptions. Suppose you have two contexts, “relatively rich customers” and “average customers”. Answer for both the following questions:

- (a) Which attributes would you choose?
- (b) Which attributes would you consider as relevant?
- (c) How is that related to the utility to and to preferences of the customers?

Exercise 2 Suppose you are in the domain of used cars. Look at the three contexts, building cars, marketing and repairing cars. Find for each context typical attributes that would not be used in the other contexts.

Exercise 3 Define a recommender system for students who want to study in your area. Formulate typical queries the students may have.

References

- Bergmann R (2002) Experience management: foundations, development methodology, and internet-based applications. Springer, New York
- Bergmann R, Richter MM, Schmitt S et al (2001) Utility-oriented matching: a new research direction for case based reasoning. In: Professionelles Wissenmanagement Erfahrungen und Visionen. Shaker, Aachen, p 264
- Burke R (2000) Knowledge-based recommender systems. In: Kent A (ed) Encyclopedia of library and information systems, vol 69(32). Dekker, New York
- Montaner M, Lopez B, de la Rosa JL (2002) Improving case representation and case base maintenance in recommender agents. In: Craw S, Preece A (eds) ECCBR 2002: advances in case-based reasoning. 6th European conference, Aberdeen, UK, September 2002. Lecture notes in computer science (lecture notes in artificial intelligence), vol 2416. Springer, Berlin, p 234
- O'Sullivan D, Wilson DC, Smyth B (2002) Improving case-based recommendation. In: Craw S, Preece A (eds) ECCBR 2002: advances in case-based reasoning. 6th European conference, Aberdeen, UK, September 2002. Lecture notes in computer science (lecture notes in artificial intelligence), vol 2416. Springer, Berlin, p 219
- Smyth B, McClave P (2001) Similarity vs. diversity. In: Aha DW, Watson ID (eds) ICCBR 2001: case-based reasoning research and development. 4th international conference on case-based reasoning, Vancouver, BC, Canada, July/August 2001. Lecture notes in computer science (lecture notes in artificial intelligence), vol 2080. Springer, Berlin, p 347
- Su X, Khoshgoftaar TM (2009) A survey of collaborative filtering techniques. *Adv Artif Intell Res* 4:1–19
- Wilke W, Lenz M, Wess S (1998) Intelligent sales support with CBR. In: Lenz M, Bartsch-Spörl B, Burkhard H-D et al (eds) Case-based reasoning technology: from foundations to applications. Lecture notes in artificial intelligence, vol 1400. Springer, Berlin, p 91

Chapter 4

Application Examples

4.1 About This Chapter

This chapter will describe some applications that are typical for CBR. We recommend it to readers who have completed reading the previous chapters. The purpose of this chapter is to provide the reader with an initial impression of the reasoning tasks and application domains typical of CBR. The application types are illustrated by specific examples. These examples are supposed to indicate where CBR can be of use and what aspects in each application are suitable for the CBR methodology. The applications will not be described fully in technical details; rather, the descriptions are intended to provide examples for illustrating the concepts and methods mentioned mostly in Chap. 2, Basic CBR Elements. Throughout the chapter, we reference where to read further about new topics. As previously mentioned, this chapter does not have a tools section.

4.2 General Aspects

The examples in this chapter are supposed to indicate where CBR can be of use and what aspects of each application are suitable for the CBR methodology. For each example, we mention the chapters for learning more about those subjects. At the end of each example we present a summary that emphasizes novel technical aspects illustrated by the example.

CBR applications can be categorized in many ways. In Chap. 2, Basic CBR Elements, we introduced multiple reasoning tasks (e.g., diagnosis, planning, and configuration). They represent one way to categorize CBR applications. Here we introduce a different one, that of synthetic and analytic tasks. A hierarchical view on application types looks as shown in Fig. 4.1. The entries are not mutually exclusive. The categories are standard when one describes varieties of computer applications.

In order to better understand synthetic and analytic tasks, we introduce a categorization of problems as being either static or dynamic. A static problem is one that is solved by a solution in one step. A dynamic problem is solved by a sequence of actions. Here it is important to add that dynamic is not the same as incremental.

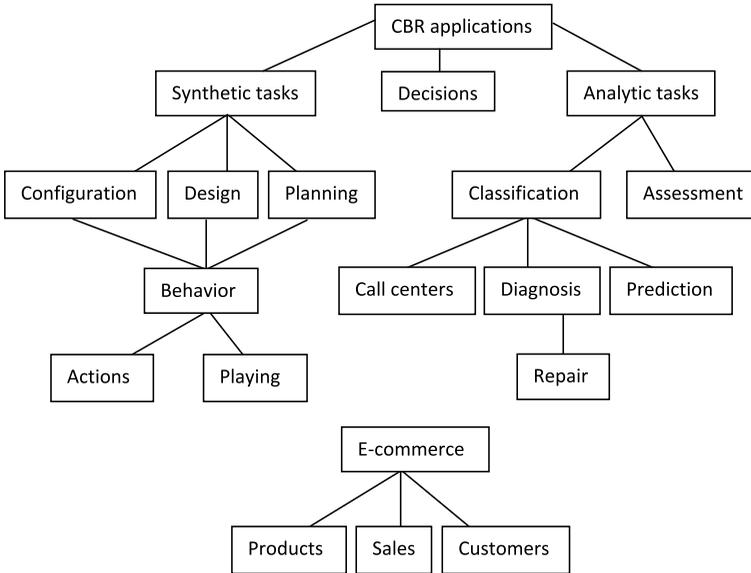


Fig. 4.1 Categorization of CBR applications

The term incremental refers to the delivery of knowledge, for instance, the knowledge delivered by cases. Even for a static problem, one often gets knowledge incrementally.

To understand synthetic versus analytic tasks, we observe their results. The results of analytic tasks are generally of static character. That means their usefulness can more or less be controlled directly.

The results of the synthetic tasks, however, are often of dynamic character, as in action planning. Often, these results describe processes or behaviour in a process. Their usefulness can, at least in many situations, be only detected by either running or simulating the process.

The complexities around the categorization as synthetic or analytic challenge their depiction as is shown in Fig. 4.1. A main reason is that synthetic and analytic tasks cannot be separated in an application. For instance, a diagnosis is a static object. In order to achieve it, often a dynamic diagnostic process is needed.

Synthetic tasks, on the other hand, need analysis, knowledge collection and knowledge understanding. They require investigation processes as well as classification and diagnostic tasks.

Some applications may have elements of both. For example, e-commerce applications usually include synthetic as well as diagnostic and planning elements. In addition, there are systems that include multiple applications. These systems typically have one main application that performs the main task users need. There are also auxiliary applications that may be needed to process intermediary steps; these are not self-contained applications but internal tasks. Figure 4.2 gives an overview. An example is the dialogue sometimes necessary in user interfaces. Another is the

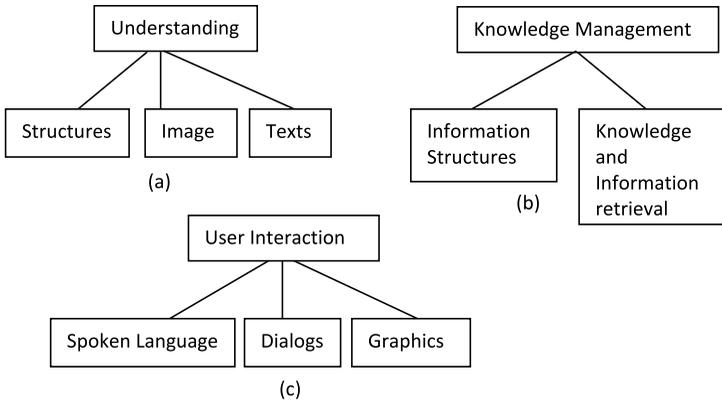


Fig. 4.2 Auxiliary applications tasks

repair of a machine. A diagnosis is a required initial step. The diagnosis is not the end of the problem solving but a necessary auxiliary task. This is discussed in some examples.

Understanding is crucial. One has to understand data, structures, complex objects, texts, or images. This can be a problem in itself that may require CBR techniques. Nevertheless, there may be tasks that are components of another task, and the latter may be the main one in an application.

Knowledge and information management is connected to all kinds of applications. Knowledge has to be collected, structured, stored in the knowledge containers, and be made available. In all areas, images and texts may also be crucial. Application issues around texts, images, sensor data, and knowledge management are discussed in Part IV in their respective chapters. Direct interactions with users may also play crucial roles in all kinds of applications. This is true, in particular, in e-commerce, where dialogues can play a special role. This is also discussed in Part IV, in Chap. 20, Conversational CBR.

The following sections describe examples organised by areas of focus where the same kinds of difficulties tend to appear either because of the reasoning task, the domain or user community, or some other aspect that gathers a family of applications. We start with analytic tasks: classification, diagnosis and prediction. Then, we continue with synthetic tasks: planning, configuration, and design. Next, we move to organisation-oriented applications: call centres, e-commerce, decision making, and knowledge management. For domain-specific applications, we discuss Law and Medicine. We conclude with examples of families of applications gathered by complex knowledge sources, text and images.

4.3 Analytic Tasks

Classification, diagnosis and prediction are all analytic tasks. They are not the only ones, but they are probably the most common ones. They are very similar to each

other. These reasoning tasks all attempt to associate an instance with a class. Classification does it in its purest form, while diagnosis and prediction are variants. Diagnosis associates an instance with a class that implies a fault, a problem, or a disease. The relationship between these outcome classes (e.g., fault, disease) and the instance are commonly provided by symptoms, and they tend to have an aggregated effect that characterises the outcome. Diagnoses are needed in medicine, all organisations, and for any machinery or artificial device. Prediction associates an instance with a class representing an event that has not yet occurred. Hence, it has a temporal dimension to it.

4.3.1 Classification

Classification is a basic application. It occurs as an aspect in many other applications. The abstract formulation is: There is a set U of objects and a set $Cl = \{C_1, \dots, C_n\}$ of subsets of U called classes. Often, one assumes that the classes cover U . The goal of classification is to assign to each object the class(es) to which it belongs. If the classes are disjoint, this class is unique, which we assume here for simplicity.

Mathematically, a classifier is a function:

$$U \rightarrow Cl.$$

Classifiers assign to each object in U some class(es). One distinguishes between crisp classes, where the membership in a class is of a yes-no character and fuzzy classes, where one has degrees of membership. Such classes are not precise sets in the classical sense of set theory. A person who is born in a country has citizenship that represents a crisp element of a class, e.g., John is a citizen of Nigeria. However, we may classify countries as small, medium and large. Then the classification of Nigeria is fuzzy. That is, the bounds that define the set of small countries are not defined. It is to some degree small and to another degree large. Fuzzy Set Theory is introduced in Chap. 15, Uncertainty.

Better understanding of the classification problem is desired when a definition for classes is not available. Many problems that are not clearly classification problems can be transformed into classification problems; this contributes to the popularity of classification among researchers. An example is predictions in weather forecasts. Predicting weather is not clearly a classification problem but it can be formulated as one as a dataset can be considered as a member of a sunshine or rain class. Here we again see fuzzy sets. because we will have more or less sunshine.

There is a wealth of classification methods, i.e., methods to classify instances into classes. Many can be found in Machine Learning. Popular techniques are, for instance:

- Decision trees for symbolic representations.
- Linear separation methods for numerically coded representations.

Table 4.1 Animal classes

Attributes	Case 1	Case 2	Case 3	New case
Animal	Sand lizard	Snowy owl	Mandrill	Opossum
reproductive characteristic	lays eggs	lays eggs	produces lactiferous meal for offspring	produces lactiferous meal for offspring
ability to grow lost parts	yes	no	no	no
classification	amphibian	bird	mammal	→ mammal

When these methods can be applied successfully, they are hard to beat. However, in many situations, they are not applicable and CBR may be a better choice. Consider, for example, subsymbolic methods where decision surfaces are generated to separate the classes. They require that the classes be linearly separable or that they can at least be easily transformed into such a representation. The k-nearest neighbour search typically used in CBR does not have such requirements.

A typical classification task is the classification of animals into birds, mammals, amphibians, and so on. A distinguishing feature among these classes is their reproductive system. Therefore, a feature would be about it. For example, an attribute reproductive characteristic would receive the values lays eggs or produces lactiferous meal for offspring. Another feature would be about the ability to grow lost parts, whose values would be no for birds, no for mammals, and yes for amphibians. Even a very small number of features can discriminate instances; hence a small set of seed instances can be further classified. This is shown in Table 4.1.

Example Highlights This example shows how attributes have to be carefully selected. Both the instances that are to be classified and the target class have to be taken into account. The main challenge is to determine the level of abstraction of the selected attribute. In the taxonomic organisation of animals, there are categories and subcategories such that some characteristics that may seem useful can cause errors. An example would be to consider having four legs as a characteristic of mammals; this would cause errors when classifying whales and bats.

4.3.2 Diagnosis

In a superficial view, diagnosis (medical or fault diagnosis) is merely a classification problem. The particular characteristic in diagnoses is that the target classes are diseases in medical problems and faults in technical fields. Furthermore, diagnoses that are limited to classification are typically useless. The only reason one needs to diagnose a problem is to identify a therapy or to repair. Therefore, it is frequently the case that a diagnosis contains two parts, the diagnosis itself and the remedy. It is also possible that the diagnosis is omitted, and only the repair or therapy is given. The example in Table 4.2 shall clarify the distinction.

Table 4.2 Explicit and embedded diagnoses

Car symptoms	Solution	Solution type
will not start, engine revs, lights working	Get some gas!	Repair with embedded diagnosis
will not start, engine does not rev, lights not working	Out of battery	Diagnosis only
will not start, engine does not rev, lights not working	Get a charge to check if battery is still functional; if it is, check alternator	Use explicit diagnosis and recommend additional step

As seen in the example, it may be easy to recognise circumstances of embedded diagnosis. It is likely that a direct repair solution is given as long as it is a one-step solution. Sometimes therapies will depend on further understanding of the problem and even can entail further reasoning tasks.

There are two main areas for diagnosis:

- Medical diagnoses
- Technical fault diagnoses

Both areas present challenges, and they will be illustrated by examples.

4.3.2.1 Medical Diagnosis

Diagnosis and prescription are very common, and the way they seem to be approached by physicians and nurses, diagnoses tend to be based on previous experiences and on evidences. These reasoning styles are consistent with CBR. Before we discuss an example, we also note the nature of the diagnosis task that is dependent upon external sources, such as exams, but is also used to interpret them. These are applications that use images, which are discussed later in Sect. 4.6.2.

The example we present illustrates the intrinsic connection between the diagnosis and prescription tasks. Although they may use distinct features, prescription may be seen as both an additional step and a consequence of diagnosis, and diagnosis can be seen as embedded in the prescription task. However, the reasoning task *prescription* differs from *diagnosis* because it is itself a synthesis task while diagnosis is a classification task.

The example we discuss is a case-based prototype developed for the diagnosis of Alzheimer's patients. However, Alzheimer's is a disease that defies diagnosis. The way to treat patients with degenerations consistent with Alzheimer's may focus on their symptoms. Consequently, the implementation we discuss is used to determine whether patients with behavioural problems might benefit from the use of neuroleptic drugs. Therefore, it is a diagnosis, because it is a binary classification task that will determine whether yes, patients have a disease that can benefit from it, or not, they do not.

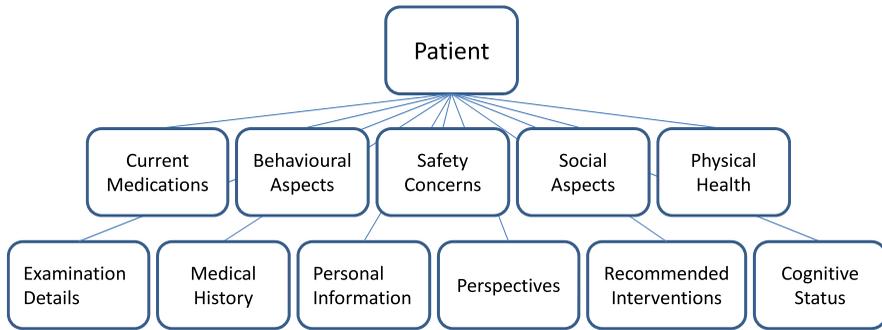


Fig. 4.3 Top layer of hierarchy of case features

Table 4.3 Resulting indexing features

agitation; paranoia; anxiety; wondering; hallucinations; caregiver stress; concurrent Parkinson’s disease or Parkinsonism; external support services; problems with dressing, bathing, or walking.

Table 4.4 Similarity function for binary attribute *wanders*

New patient	Patient n	Sim (new, n)	Patient $n + 1$	Sim (new, $n + 1$)	Patient $n + 2$	Sim (new, $n + 2$)
Yes	No	0	Yes	1	NA	0

The physician’s assessment of the features used in this decision led to about 100 features. Those were organised hierarchically. Figure 4.3 shows the top level.

With a case description, available information can be used to describe real cases. The next step is then to consider how to assess similarity. From among so many features, which ones may be the most relevant, in order for the resulting retrieval to produce the most similar case for the right decision to be reused?

Table 4.3 lists the eight features used as indices. They are weighted equally in nearest neighbour matching. Some of the features are binary; others have a range of values.

The local similarities were established as follows. An exact similarity, match or not, was used for binary features. Table 4.4 illustrates three situations when comparing a new query problem, a new patient, with three other patients. Whenever one of the cases being compared did not have a value for a feature, that feature was considered not to match with that case. Degrees of similarity were used for features with a range of values.

Example Highlights This example demonstrates a reasoning task diagnosis that, at first glance, might seem a prescription task. It illustrates some of the issues with many real-world problems where a large number of features have to be carefully studied and categorized. From 100 features, the similarity measure eventually used only nine indexing features. The example also shows a decision with respect to missing values.

4.3.2.2 Technical Diagnosis

Consider the fault diagnosis of a machine that has several components. The existence of a fault can be detected by humans in manufacturing and service situations where humans can easily observe faults. In industrialized settings, systematic maintenance is typically put in place to recognise faults.

The first step for building a system is understanding the problem informally. This means knowing what the possible problems and solutions are, which requires grasping a few concepts:

(1) About the machine:

- a. There is a direct subcomponent relation and the part-of relation is the transitive closure of it;
- b. There are ports that constitute the only connection of a component to the outside, i.e., to other components.

One has to know the basic elements for these descriptions.

(2) About the problem formulation:

- a. The problem is defined in terms of observations and these are reported by the ports. The observations may have a static or a dynamic character. The latter is about behaviour. Faulty behaviour describes the behaviour of the component if certain faults are present, as when something is too slow. The function description describes the behaviour of the whole machine rather than the behaviour of components. It uses the port variables which connect the machine to the outside world. It is usually too complex to describe all input-output relations, and therefore is restricted to the intended behaviour of input ports of interest.
- b. Component influence describes the influence of a component on other components, such as gas going to the engine. For instance, certain input ports can determine the values of certain output ports.

(3) About the solution:

One has to know the structure of the machine in order to interpret the diagnosis correctly. There is a huge variety of types of technical faults. We will just present one example that covers several aspects.

In the example of diagnosis of faults of a CNC (computer numerical controlled) machine, problems and solutions have to be formalized. This contains a description of the components of the machine as an essential part. An example of the description of a component in a CNC machine is:

```
PrimitiveComponent define: Relay
ports: ((currentIn dc in medium) currentOut dc in medium)
       behaviour: (((currentIn = +) (currentOut = -)
                   → (lever = shifted))
                 ((currentIn ≠ +) → (lever = unshifted)))
```

Table 4.5 Example of a case

Problem, observations	Value
Error message	I59
I/O state OUT7	Logic 1
Valve 5Y1	Shifted
Valve 5Y2	Not shifted
Control system	Working
I/O state IN32	Logic 1
Acceptance cone	Clean
Diagnosis	
I/O card	Defect

((currentOut \neq -) \rightarrow (lever = unshifted)))
 failures: ((stuck (lever = shifted))
 (magnetDefective (lever = unshifted)))

Now we are in the position to formulate cases. An example is in Table 4.5.

These observations are recorded in the case base. In principle, one does not know if they are complete and/or redundant. In a good case base, they would be complete. For building the case base one has to refer to recorded experiences. In addition, one can invent artificially new cases if certain areas are not covered sufficiently.

The next step is the development of the similarity measure. At first, assume that the problem attributes are independent. This allows us to use a weighted similarity measure. The local similarities for this kind of machine are not very difficult. Many domains are Boolean or have numerical values where the similarities can easily be estimated.

The determination of the weights is more complex. This will be discussed in more detail in the Chap. 6, Basic Similarity Topics.

Once local similarities and weights are determined, we obtain the nearest neighbours. It is then necessary to decide if the solution seems to be reliable enough in order to be accepted. This is done by looking at the similarity degree $\text{sim}(\text{old}, \text{new})$ between the old and the new problem.



For this purpose we choose thresholds τ , $0 < \tau < 1$. If $\text{sim}(\text{old}, \text{new}) \geq \tau$, then the old case is accepted because it is sufficiently similar to the new one. This is extended to the method of rough sets in Chap. 6, Basic Similarity Topics.

Example Highlights The description of the machine becomes feasible once its components and their relations are understood. There are visible elements called ports where observations can be made for describing the problem. Both description and

similarity follow the local-global principle using the part-of relations (the components); the local-global principle will be introduced in Chap. 5, Case Representations. There are thresholds for accepting the solution.

4.3.3 Prediction

Predictions deal with events that will or may happen in the future. Predictions are more precise than general forecasts. However, often predictions can be reduced to classification. Several methods apply classification to predictions. This is because histories in the same class are similar in the sense that they share the same predictions. This is, for example, the case if a prediction deals with a fixed number of alternatives. Each alternative gives rise to a class of situations where the prediction is valid. Therefore, the main task is to detect this class.

In general, predictions deal with dynamic situations. A case is of the form

(history, prediction).

In a new situation one searches for the nearest neighbour in the histories and uses it directly or after some adaptation. The prediction can be symbolic or contain numerical data. The essential element is the identification of an appropriate similarity measure.

There are many prediction problems of commercial relevance in software engineering. Two examples for software projects are estimating the number of defects and estimating effort.

A major distinction for predictions is with respect to time:

- (1) Time is not explicitly represented in the following:
 - a. If we perform this, is it likely that an error occurs?
 - b. If a patient gets some medication, will it help?
 - c. How costly will this procedure be?
- (2) There is an explicit representation of time and historical data; this includes time series analysis:
 - d. Weather forecast
 - e. Currency exchange rates

Often, more general problem situations are called prediction problems because one does not (yet) know the solution. However, it is more correct to calling a statement a prediction if time is involved and one can talk about the future. This has an effect on the evaluation of the solution: It cannot be done immediately but only after some time has passed. Therefore, the two examples a. and b. in (1) are strictly speaking not predictions.

In the following example, time is not explicitly involved. We want to predict the needed effort for a software project. The first step is to understand the problem. The

effort is defined in terms of hours for the participating persons. A case is a past project of the form (project description, effort). In order to define precisely what a case is, one has to find out the relevant characteristics of a project with respect to effort and formulate them in terms of attributes that attempt to cover all the needed information for the prediction. Examples for the attributes are:

- Languages = {C++, Java, VB, JavaScript, VBScript, SQL, Php, Perl, ASP, HTML, XML, others}
- Database systems = {Oracle, Access, SQLServer, MySQL, others}
- Methodology = {OO, SA, SD, RAD, JAD, MVC, others}
- Complexity of internal calculation = {very low, low, medium, high, very high}
- Language and tool experience level, number of months of experience = positive integer
- Number of data files accessed = positive integer
- Number of data entry items = positive integer
- Number of data output items = positive integer
- Team size for implementing the object = positive integer
- Functionality is a vector of numbers showing the percentages of the activities (1-Internal process, 2-Data entry/Modification/Deletion, 3-Output form (screen), 4-Data query from database/file, 5-Printing, 6-Report, 7-others).

The attributes define what a case is. For a case base we need many previously recorded experiences that a company usually has. The structure of the company and the work it does most often influences the handling of the attributes in different ways:

- The attributes may not be completely independent, for instance, the experience level can depend on the language.
- They may influence the definition of similarities. For some companies the influence on the importance of an attribute may be different than that for other companies.

The similarities between attributes, the local similarities, are generally easy to define. One feature that may pose a challenge is the last one, the functionality vector. The difficulties in determining the weights in a measure are discussed in Chap. 6, Basic Similarity Topics. As a first approach in an iterative process, using equal weights for all attributes may be justified.

Finally, another difficulty occurs in the evaluation of the results. Here, one has to define what the accuracy of a prediction means. This problem boils down to deciding when predictions are seen as sufficiently similar. This in turn is a consequence of the goals of the company as well as of the quality of the similarity measure. Such problems are typical in software development.

Example Highlights Time can be implicitly embedded in a prediction task. Predicting software development effort is difficult for many reasons, but mostly because of the many aspects one can associate with software projects. It is not clear what the vocabulary is, what is important for the effort and what to include in case representation. The importance of a case attribute may depend on the organisation the

application aims to serve. The evaluated quality of an application depends on the client too. This is not an easy task mainly due to the approximate nature of CBR. On the other hand, other methods face the same problems. One can learn from other methods much about representation and evaluation issues.

4.4 Synthetic Tasks

We consider three major types of synthetic tasks:

- Configuration
- Planning
- Design

They have in common that for a certain purpose a new complex object is created, mainly from simpler objects due to a number of restrictions.

Configuration is in some sense the simplest task. It relies on a fixed number of objects. They may have parameters (such as size) and constitute the parts of the object to be configured, typically a technical device.

The next type is planning. The result is typically a dynamic object that controls behaviour and tells what one has to do.

Design has elements of both of the other types. The purpose is to obtain an object with a certain function where one has a large degree of freedom.

4.4.1 Configuration

Given a set of components, configuration is the task of selecting a subset of components and choosing their parameters to specify a system (e.g., a technical system). Understanding configuration requires understanding the needs of the system user as well as the system to be configured. The difficulty arises from the many constraints when selecting components and from the specific application.

Consider a shop that configures bicycles for special customers from prefabricated parts. These parts can be combined for developing the final product. In addition, there are adaptation rules for changing components according to certain constraints. The case base is constituted of configuration plans for

- Whole bicycles
- Components

In addition, there are certain rules that allow exchanging components. They establish the rule container. We use informal description elements, but they can be easily expressed formally using attributes.

Customer description examples:

- Specific type of use: Triathlon, driving to work, driving in the woods and so on.
- Average intended speed.
- Static body measurement metrics: Torso length; arm length; shoulder width.
- Price.

Some bicycle elements:

Whole bicycle:

- General bike type: Racing, mountain bike, others.

Weight components: Frame; wheels; saddle area; front set; pedal; rear brakes; chain rings.

(1) Frame geometry:

- Saddle height, the distance from the centre of the bottom bracket to the point of reference on top of the middle of the saddle.
- Reach, the distance from the saddle to the handlebar.
- Drop, the vertical distance between the references on top of the saddle to the handlebar.
- Setback, the horizontal distance between the front of the saddle and the centre of the bottom bracket.
- Standover height, the height of the top tube above the ground.
- Toe overlap, the amount that the feet can interfere with steering the front wheel.

(2) Frame materials: Steel; aluminium alloys; titanium; carbon fibre; thermoplastic; magnesium.

In addition, there are many constraints between the different possible components that are left out here; for instance, as bicycles get larger, stems get longer, and vice versa.

The general configuration strategy uses the hierarchical ordering of the components. In principle, we have the following steps:

- (1) Select a configuration plan that results in a product similar to the demanded product.
- (2) Analyse the components:
 - a. If the component is sufficiently close to the demand, keep it.
 - b. Otherwise search in the case base for other configuration plans for this component.
 - c. If the similarity to the demand is increased, check the component's compatibility with rules and accept it in the positive case. Often, the similarity is increased if the price differs.

First we consider the bike as a whole. For this we have to define the attribute-value:

- Attribute-values of “specific type of use” and “general bike type”;
- Values of price;
- Values of weight and speed.

We observe that many specific attributes cannot be considered on this level. For this reason components have to be checked. As an example we consider:

- Static body measurement metrics
- Frame size

A first advice for defining local measures is given by guidelines like, “Frame size was traditionally measured from the centre of the bottom bracket to the centre of the top tube”. Typical “medium” sizes are 54 or 56 cm (approximately 21.2 or 22 inches) for a European men’s racing bicycle or 46 cm (about 18.5 inches) for a men’s mountain bicycle. This has to be specialised by taking the measurements of the customer into account.

There are many technical constraints between components. The components have ports as introduced in the diagnostic example and the main constraints are between the port values. For example, certain wheels, saddles, or pedals are incompatible with certain frames due to the fact that power on the pedals will ultimately be transformed into the speed of the wheels. This will give the reader an idea of the level of complexity of the constraints.

Example Highlights First, we observe that the vocabularies describing the customer and the products (the bicycles) are almost disjoint (compared to when selling of houses, for example). This increases the task of defining a similarity measure between the demand and the product to be configured.

Second, we observe that a top-down strategy for making configuration plans was used. This is explained in more detail in Chap. 9, Adaptation, where techniques using several cases are discussed.

The attributes in this example are twofold. They contain specific parts and their properties as well as background knowledge. Background refers to domain-dependent knowledge, for example, when a bicycle is to be used for triathlons, which requires understanding of this kind of use. The need for background knowledge is an opportunity to use either humans in the process or a knowledge representation structure such as a base ontology (for ontologies see Chap. 12, Advanced CBR Elements). An additional knowledge source would allow the integration of a subsequent system like fault diagnosis, for example.

4.4.2 Planning

Planning problems are particularly suited for CBR because plans are often not created from scratch but are rather variations from previous plans. The distinguishing elements of planning are states. A plan is a sequence of actions that modify an initial state to reach a goal state. Plans consist of a number of action steps; this is described in detail in Chap. 5, Case Representations.

Cases should include the problem where a state is described. The attributes used in a case representation have allowable values and their combination determines a state. The solution part of the case is the actual plan, that is, the set of actions that, once executed, will change the values assigned to case attributes, thus changing the state. Cases for case-based planning should also include an outcome element

to compensate for the lack of general knowledge. For example, an adapted plan may have caused an unforeseen error. This may give rise to introduce a new case. Case outcomes may also include plan provenance, which is discussed in Chap. 11, Development and Maintenance.

The provided problem that needs to be solved is that of making a plan to deliver products in a city in the shortest time possible. The information is available on an electronic map. Most online maps today include information about traffic conditions and constructions, but they are usually slow in updating information and they do not embed experiential knowledge about alternate routes and how those may also be impacted by traffic patterns.

The case base comprises route planning experiences that entail aspects that are related to the problem, such as origin and target destination, time of day, day of the week, that tend to repeat; in addition to other less common aspects, such as a special event in the surrounding area, construction, holiday traffic variation, weather, and so on.

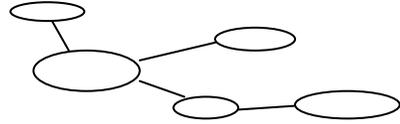
The first problem is to decide how to represent cases. A flat representation using attribute-value pairs would fail to benefit from the graphic representation in a map. Above all, the planned route needs to be feasible in the map, so a case representation needs to use a map. Furthermore, one has to determine what the scope of a case will be. Should a case comprise a complete route or should routes comprise a concatenation of cases? What this determines is that in a space of solutions where cases and experiences may not map exactly to each other, an additional representation of the problem domain is also needed. In this example, deciding about the representation of the map, including the ability to modify it as experiences are learned, is as important as deciding about cases.

A usual problem is the process of case acquisition. Given that these are meant to be experiential cases, they are to be acquired as a user lives an experience. Consequently, there will be portions of the solution space, that is, areas in the map, for which experiences will be available, while for other areas or routes no experiences will be available. When the system is to create a new route by reusing previously recorded experiences, what would the potential benefits be of reusing a previously known route over selecting a yet unexplored one?

Once all these aspects of case and domain representation are considered, the system has to be designed to use a new given problem to create a solution. The new problem consists of the elements of the route, such as origin and goal destination, but may include intermediary destinations as well. This would require a multiple step creation of the route. All other available information such as day of the week and time of day are useful to help better utilize recorded experience. The benefits of reusing known routes are in the additional experiential knowledge the experiences include that can be used to determine whether one route can be better than another. Figure 4.4 shows the map.

Consider the new problem where a user who wants to go from the George Washington University to the White House in Washington, D.C., on a Tuesday at 3:00 P.M., using public transportation. Let the only known routes be the ones that can be taken on foot or by bus. Also, assume that we have given a parameter to the

Fig. 4.6 Cases graphically described



4.4.3 Design

What distinguishes design from other synthetic tasks is the character of the features in a design case. First, design features should not be perceived as decomposable. Even if in reality they could be decomposed, design is an ensemble of features that complement each other for a common purpose. Individually, each feature has a function, such as windows allowing ventilation, lighting, and views, or the positions of chairs the directions in which people will be looking. But it is the impact of the ensemble that determines the quality of the design. The most important aspect in design is reuse. Yes, adaptation seems to be equally important, as rather than starting designs from scratch, case-based design allows designs to be reused and adapted to create new ones.

Considering the background of the users is of utmost importance. What we are trying to emphasize is that there are some applications whose users can be from a specific profession (e.g., architects, lawyers, physicians) that strongly characterise them in terms of the way they think or the kinds of objects they are used to interacting with.

We describe a system where users are architects who are used to manipulating graphical objects. As a result, cases are described graphically through modifiable diagrams. These diagrams show major rooms and some properties and relations but not the absolute physical positions, which are part of the final solution. They look as in Fig. 4.6.

Such a representation is called a bubble diagram because the nodes are represented as bubbles. It represents the original demands and is easier to handle by an architect than text.

Each case is a complete design of residential units, not individual rooms. Developers justify this choice because in architectural design, the context in which a room design occurs is more important than the individual design of each room. Cases are organised through multiple representations by attribute concepts. Each residential object has two sets of attributes:

- Simple attributes like
 - area, number of spaces.
- Complex attributes like
 - rooms zone, services zone, circulation zone, open spaces zone, connectivity.

Complex attributes are in fact other objects that have different attributes themselves. For example, rooms zone is an object with two simple attributes (area and

number of spaces) in addition to five other complex attributes representing five spaces that this zone might include. Each space is represented by four simple attributes (area, position, orientation, and space type).

House is at the top-level, connected to Zone, which has the concepts Room, Service, Circulation Zone, and Open Space Zone as children concepts. Each of these concepts encompasses a number of attributes, which are valued in each case; in this example there are some specific difficulties with the attributes.

First, there is no guarantee that the rooms within the *roomsZone* are ordered in the same exact way in both the query and the case, even if they contain the same room types. For example, the query might include a master bedroom in *space1*, a secondary bedroom in *space2*, and a dining room in *space3*, while the case includes the same rooms, but in another order (secondary bedroom first, then the master bedroom and the dining room).

Second, the query might contain room types totally different to the case it is compared with. For example, the query might contain a master bedroom, a dining room and a study, while the case contains a master bedroom, a secondary bedroom and a family living room.

These inconsistencies create problems in comparing queries and houses, which result in difficulties in defining an adequate similarity measure. The case library organises cases in an object-oriented structure, where each case is an object. A distinguishing aspect is the need to organise the rooms into four zones to allow the most possible comparisons between rooms.

Example Highlights Gaps between the queries and the houses are characteristic of design tasks. The query comes from a buyer. This person is not mentioned here and the query goes directly to the architect, who has to select partial designs from the case base and combine them.

As we saw, the description elements are not the same as those of a naïve house would be. This is further discussed in Chap. 3, Extended View, where product descriptions and functionalities are compared.

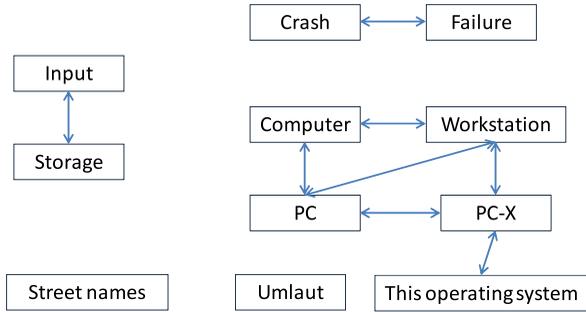
4.5 Organisation-Oriented Applications

The class of systems in this section is characterised by the organisation or field they are supposed to serve. Rather than being developed centred on a reasoning task, they gather similar tasks that are common to a department or organisational sector. In this section we discuss call centres, e-commerce, knowledge management, and law.

4.5.1 Call Centres

Call centres are centralized customer service units that use computerized solutions to manage large volumes of phone calls. In a typical operation, a human

Fig. 4.7 Similarity between words



or computerized representative communicates with customers who have problems and questions. Sometimes the first response is done automatically and a human operator (who again can be supported by a computer) is called only if necessary.

An example is the method of Frequently Asked Questions (FAQs) with which a human operator who can answer many questions immediately. Answers that are more complex are based on recorded experience. A characteristic of this situation is that a dialogue with the customer is involved, which points to conversational CBR (see Chap. 19). A standard situation for a call centre is support in the after-sales phase: I bought a device and it is not working properly. Such a conversation is typically performed in natural language. The call centre will now be supported by computers. Often, the human operator does the talking. In an extension, the communication can at least partially be done by computers too. The general communication problems of these aspects are discussed in Part IV.

Consider a call centre for dealing with personal computers and related problems. In this case, the centre has error documents stored. These documents constitute the case base. The task is to find that document that is most similar to the query.

A typical query would be, “On my *PC* the *input* of long *street names* leads to a *crash* with the message *storage error*”.

Suppose there are three possible answers in the error documents:

- A₁: Under Windows 3.1 there is not enough memory allocated for the *name* of the *street*. This may lead to a *failure* of the system.
- A₂: The PC-version does not perform a correct storage of *street names*.
- A₃: On PC-X the *input* of an *umlaut* leads to a *failure*.

For explaining the problem we mention the major steps to be taken. Ultimately, one has to relate the answers to the query. This is done by relating certain words to each other. Therefore, first some keywords are isolated. In the example, these are the words in bold. Next, the keywords in the query have to be compared with the keywords in the error documents and their similarities have to be computed. This falls in the area of textual CBR. In our example this looks as in Fig. 4.7. These similarities reflect the meanings between the keywords; these are the local similarities. The relation is the so-called quasi-synonym relation in linguistics, which means their meanings are qualitatively similar.

The analysis now leads to:

- Answer 3 is immediately excluded (because of **street names** versus **umlaut**).
- Answer 1 is the nearest neighbour because this is a frequent cause. We observe that this is knowledge contained in the similarity measure. This answer is given to the customer.

What has to be done for realising such a help desk system formally?

- (1) First there has to be a file with error documents.
- (2) One needs a vocabulary. In this case it is a list of keywords.
- (3) One needs a retrieval function that extracts the keywords from the queries and the possible error documents key.
- (4) For defining the similarities there has to be a thesaurus for synonyms or quasi synonyms.
- (5) Weights can be determined in different ways. A simple way is to count how often a pair of keywords occurs in a conversation.

Finally, the measures of success have to be discussed.

- How many repeated calls are avoided?
- Is the average time for a call shorter?

Example Highlights The aspect that distinguishes this application from the previous ones is the representation of queries and answers. Instead of using attributes and their values, one uses ordinary text. This is the topic of textual CBR in Part IV. In addition, the queries are not formulated completely but rather completed in a conversation. This is discussed in Part IV, Chap. 20. The example was simple in several respects. Mainly, there were no long conversations and only simple error documents. Nevertheless, some essential aspects have been illustrated.

4.5.2 E-Commerce

In e-commerce as in any other form of commerce, the customer has a demand. Here we are interested in the situation where the demand can be satisfied to some degree only and where the demand can be specified by the customer only incompletely or inexactly. The first aspect points again directly to CBR while the second one concerns conversational CBR as discussed in Chap. 20.

4.5.2.1 E-Commerce Example

Now we consider an e-commerce shop for selling houses. We focus on houses for young professional couples.

Again, we start with understanding the problem informally. This means two views that are not independent: Understanding the customers and understanding

the houses. We start with an informal description. This will lead to an introduction of the attributes and their importance.

The young couples may have plans to have children in the near future. Therefore, they take into consideration their job situation as well as facilities for toddlers, for example. Facilities for adolescents will be of less importance to them. These things, among others, influence the desired environment, the location of the house, and the number of rooms.

For modelling a house for the customer class, the following steps have to be performed.

1. Identify an appropriate high-level case representation.
 2. Introduce attributes and types that apply to desired houses for the chosen class.
 3. Define relevancies of attributes to the customer class in terms of weights.
 4. Define local similarity measures.
 5. Define appropriate Completion and Adaptation Rules as required (see Chap. 9).
 6. Build a case base.
- (a) For describing the CBR system we have to transform the informal descriptions into formal models. This is done stepwise.

Here we have to realise that for the purpose of sales, the view of an architect is not defined. In fact, the model has to consider aspects that have nothing to do with the house as a building, such as the social environment. Identified as major components of the house description are the following concepts:

- Professional couple house
- Details of the house
- Other possible additions to the house
- Access to social amenities

(b) Attributes and types.

The range of attributes is usually chosen as subtypes of standard types. One example is the *House Location* type called *Location Area*, a subtype of Taxonomy Symbol. We describe this in Table 4.6.

One can understand location to be hierarchical, such as geographical partitioning of the city. With this definition, it is easier to get into every detail of location a buyer might desire. Examples for subclasses of location: outskirts and other city area. Under these subclasses, we have detailed definitions identifying whether a customer needs northeast, northwest, and so on, of these subclasses. After choosing the principal location, a customer also has the flexibility of going to a further level of detail, specifying whether she wants a place, a street, a road, and so on. This leads to a formal case description as shown in Table 4.6.

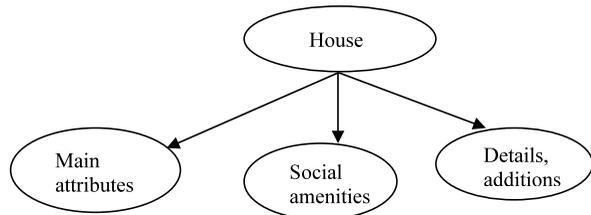
Additional (virtual) attributes are:

- Price per sqm;
- Total real house cost = Price + brokerage + survey costs.

Both are not absolutely necessary but are helpful for the customers.

Table 4.6 Formal case description

Attributes	Types	Domain
House Type	HouseType (Symbol)	Duplex, Condominium, High-rise, Family House, Ranch, Studio, Townhouse
Social Amenities	SocialAmenities	Reference
Price	HousePrice	Real
Rooms	RoomNumbers	Integer, 1–5
Bathrooms	Small Integer	{0, 1, 2}
House Details	HouseDetails	Reference
Details of Additions	AdditionDetails	Reference
Age of House	Small Integer	{0, 1, 2}
Location	Location Area (Taxonomy)	{suburbNorthEastOnRoad, DownTown, suburbSouthWestCrescent}
Payment Terms	payTerms (Symbol)	{Mortgage, full payment}
Mortgage Broker Fees	HousePrice	Real
Survey Cost	HousePrice	Real
Availability Date	AvailabilityDate	Date
House Area	Area	Integer

Fig. 4.8 Abstraction of house description

Some of the types are reference types. That means they refer to other types. We mention these references briefly:

Social Amenities: Nursery, schools, shopping centre, public transportation, sport facilities. These details have only to be mentioned once if one has many houses in stock that are located in the same area.

For details we have: Kitchen facilities, car park size, balcony, heating, air conditioner and additions such as swimming pool.

The advantage of these reference types is that the customers need not see the details in the beginning because they have little influence on the decision. The structure of this house description is shown in Fig. 4.8.

We have also to define appropriate similarity measures for all the subtypes defined. For example, suppose the query states *DowntownAvenue* and the retrieved case contains *Downtown*. The resulting similarity score should be lower than the score given when the query states just anywhere *Downtown* and the case contains *DowntownAvenue*.

Fig. 4.9 KM organisational process



Since price is highly important, the type of similarity defined for price should result in a higher similarity score if price is lower in the lower. This is discussed in Chap. 6, Basic Similarity Topics. For the determination of weights there is no formalism for this kind of problem. It is done according to the opinion and experience of the store personnel.

When a new (catalogue) house is under construction, adaptation plays a major role. It happens frequently that a house does not match the query demands but one can easily modify the house for achieving a better match. A typical adaptation rule is used for a situation where we have a query requiring more bathrooms than available, but the retrieved case contains more rooms than the customer wants. This is discussed in Chap. 9, Adaptation.

Example Highlights The major point in e-commerce is to combine the customer and the seller views in the description of the sales objects. In this example, it was fairly easy because everybody knows what a house and its environment is. What one does not know is the individual preferences of the user what has to be modelled.

4.5.3 Knowledge Management

Knowledge management (KM) encompasses initiatives that support the rational allocation of knowledge assets from the perspective of, organisations, and computer systems. KM is typically implemented through knowledge processes that include tasks such as create, distribute, and reuse knowledge. Chapter 21, Knowledge Management, is entirely dedicated to this topic.

A KM application that uses CBR is of the type implemented over a repository. The repository turns into a case base. Repository-based KM systems are organisation-oriented and implement a KM process of the type depicted in Fig. 4.9.

The suitability of CBR as the underlying methodology for implementing these KM systems originates from the affinities between the two processes. Therefore, the main concern in such an implementation is characterising the records in the KM repositories as cases. The central record in a KM repository is a knowledge artefact.

It can take different forms depending upon the type of organisation. Some examples are best practices, lessons learned, and alerts.

A typical problem in KM is to reuse collections of lessons learned stored in a lessons learned repository. As mentioned earlier, lessons learned are knowledge artefacts that humans learn through experience. This is ubiquitous in large organisations such as multinational companies or the military.

Large organisations have pioneered the collection of lessons learned systems throughout the world. A particular case is the military. This category of KM effort is particularly beneficial to such an organisation because of the variability of contexts. In particular, military doctrine is rigid and is not meant to change with experience, making experiential knowledge of utmost importance for success.

An organisation's members are usually asked to describe experiential lessons after working on their functions. These lessons are collected because the entire organisation can widely benefit from knowing about them; they are meant to teach a lesson to anyone who would do similar tasks. These experiences can help them approach problems more intelligently, and avoid serious problems that include ensuring their own safety.

The format in which lessons learned are collected is not always proper for use in CBR. When this happens, we need to take a repository of lessons learned and define all knowledge containers to reuse lessons.

The vocabulary container includes the knowledge necessary to use lessons as cases. It comprises a high-level model for lessons based on the basic structure of a case, that is, a problem and a solution. Its attributes meet the requirements for knowledge sharing from KM. The problem characterises where the lesson applies. This includes the indices to guide retrieval.

Representing lessons as cases also illustrates the suitability of CBR in that a lesson context does not have to be exactly the same as a new context to be applicable, and it may even be partially applicable. Suppose an organisation will conduct an operation in country *X*; thus they are potentially interested in lessons that describe the applicable problem in the context of *X*. If they will conduct such an operation using helicopters, they are not likely interested in lessons describing operations on foot. Being in country *X* and using helicopters are preconditions that have to apply for the lesson to be useful.

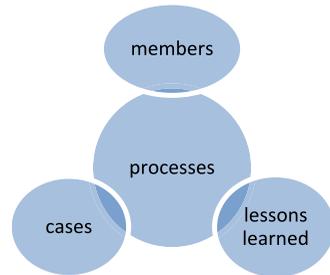
The specific task or process must be included. If the lesson describes a rescue operation, then it may not be relevant to a fuelling operation. The solution part of the lesson is the suggestion or strategy the lesson entails. Instead of an outcome, lessons include a rationale, which explains how the lesson was learned. This serves both to motivate and to substantiate the validity of the lesson. Table 4.7 shows these fields and gives an example.

Active distribution is when the KM systems, as is usually advised, are implemented as integrations into other systems that an organisation's members tend to use on a daily basis. This form of monitored distribution keeps track of processes in which members are engaged and distributes applicable lessons when they become relevant.

Table 4.7 Example of a lesson represented as a case

Case problem (indexing elements)	Applicable task Preconditions	Assign air traffic controllers Civilian airport, military air traffic
Case solution (reuse elements)	Lesson Rationale	Assign military air traffic controllers Type: Failure What? Military traffic overloaded civilian controllers. Why? The rapid build-up of military flight operations at Mactan International Airport quickly overloaded the civilian host nation controllers.

Fig. 4.10 CBR implementing KM cycle for general processes



The last field, as illustrated in Table 4.7, has subfields. This is to account for lessons that originate from failures and successes. Then, the subfield “What” describes what failed or succeeded. The type may also be an advice given by an experienced higher ranked member, so military members will have to describe that as well.

Figure 4.10 demonstrates how CBR implements a KM cycle for general processes, such as military operations. The distribution of these lessons can be done either through search or by active distribution. Active distribution, as mentioned above, keeps track of processes in which members are engaged and distribute applicable lessons when they become relevant. The final effect does not change. Regardless of the distribution method, the planner of an operation where the example lesson in Table 4.7 applies would be informed of the need to bring along military air traffic controllers to the operation.

Example Highlights This example poses challenges because of the domain-specific indexing it requires. The only way to retrieve applicable lessons is to index them according to the organisation’s target processes. Moreover, the indexing and similarity measure need to properly recognise those target processes whenever and in whatever form they are mentioned.

The quality of the repository is decisive in automating the indexing. Ideally, lessons are to be collected directly into the case target format. This will be further discussed in Chap. 21, Knowledge Management.

4.5.4 Law

Law is a special professional domain that can widely benefit from automation of reasoning tasks. CBR can be used in multiple forms in this field. In general, their vocabulary is specific and well bounded, making it hard to humans, but not so hard when a domain ontology is available. Nevertheless, both domains represent deep and complex areas of specialisation where understanding can be quite a challenge to both humans and computers. However, it is not immediately clear how to formulate law problems in a way such that CBR can be of use here.

If we consider the potential of CBR to support legal activities, then we need to examine it beyond legal reasoning. Mediation, sentencing, and outcome prediction are important and frequent legal tasks that can be supported by CBR. Mediation is considered as a design task, sentencing is mostly about adaptation. Other important legal tasks are search and interpretation.

Interpretation is not unique to the legal domain, but that is where it is more relevant for practical applications. The legal profession is strongly based on legal argumentation where precedents, or cases, are used in support of arguments. This makes CBR amenable to legal applications; both are based on previous cases.

It is when deciding whether a case is adequate to support an argument that interpretation is performed as a reasoning task. In legal CBR, previous interpretations are used to guide the interpretation of a new case: this is called interpretive CBR.

The problem context is to find analogous decisions in legal precedents in order to create an argumentation for a lawsuit. All courts publish reports where lawsuits and other legal processes are described together with their decisions. These decisions represent vast repositories that embed legal knowledge.

In order to use legal precedents to create an argument for a lawsuit, a few concepts we need to be understand. In this example, we discuss litigations between two parties, the plaintiff, who is the author of the suit, and the defendant, who is being sued. Consider a young adult who was conceived with the use of donated sperm. This young adult now wants to know the identity of her father and thus files a lawsuit against the sperm bank from where the sperm was obtained. The bank says it will not reveal the identity because of binding contracts with sperm donors on the confidentiality of their identities.

An argument in favour of the defendant is that she needs to know the identity of her father because she is seeing a man who was conceived in the same way with sperm from the same bank. An argument in favour of the plaintiff is that DNA can be used to tell whether they are siblings. The list of arguments can be long.

In order to use CBR to automate the problem of finding decisions, from a repository of legal decisions, we have to fill knowledge containers to solve our problem. The problem of searching for arguments is more difficult than that of searching for applicable lessons. For applicable lessons, we need to match the target organisational process and preconditions from the lesson and to the problem context. Finding arguments has two subproblems. First, one has to identify an argument as applicable to a problem situation. Second, one has to determine whether such an argument is

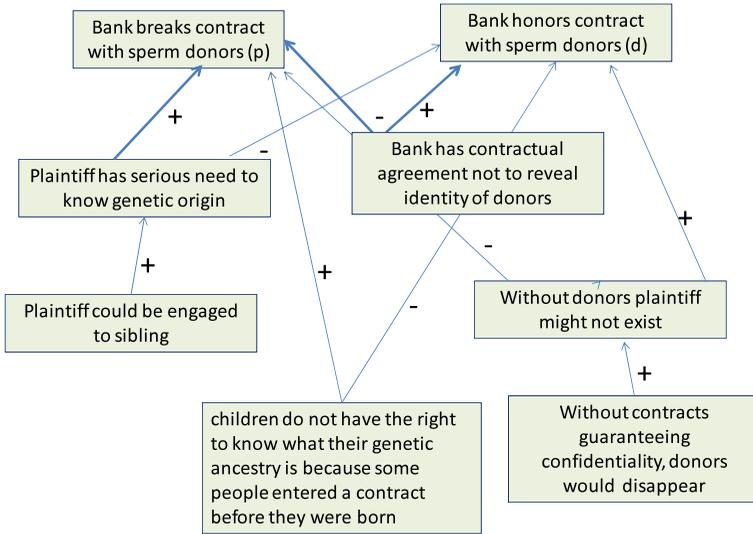


Fig. 4.11 Hierarchy of factors

in favour of or against the problem situation. Furthermore, simply associating an argument as applicable to a problem context can be very difficult.

The way to index legal precedents as cases is by identifying facts and associating them with arguments. These facts are called factors. Facts in a lawsuit become relevant to an argument when they shed light on an issue.

The indexing for retrieving arguments is better represented hierarchically because factors are interconnected. Figure 4.11 illustrates a partial factor tree. This is an intermediary representation used with a similarity measure to enable the comparison between concepts, which can be quite abstract.

Note that the illustration is a simplification of an actual tree from a legal subdomain. Actual trees would be more general and not specific to the example we are describing. For example, instead of sperm donors, the factors in the tree would refer to contractees or third parties. Trees need to be built for each subdomain, but then they can be reused in multiples problems.

The two high-level conclusions at the top of the tree in Fig. 4.11 represent the main conclusions that will determine who wins the lawsuit. The one on the left favours the plaintiff and is indicated with (p); the one on the right favours the defendant, and is indicated with (d).

Factors that support or strengthen a conclusion are connected to higher-level concepts and conclusions with arrows and the plus sign, “+”. The minus sign, “-”, is used when the association is negative, that is, the factor weakens the conclusion. The thickness of the arrows indicates their relative importance. Thick links are stronger supports for the conclusions that have plus signs.

The tree is used to compare a new problem with existing decisions in a legal case base. One particular aspect of reusing previous legal cases for argumentation is

that not only similarities but also differences between the new problem and previous cases need to be assessed in order to select an argument.

Example Highlights Indexing needs to associate statements in legal precedents with arguments and indicate whether the statement is in support of or against an argument. A special representation for indexing is required before cases can be reused.

Both similarities and distinctions between facts of a new problem and factors in the indexing tree need to be assessed. The need to both compare similarities and contrast differences is a trait of legal reasoning. An argument can be made stronger when it is compared to a similar case with the intended result its differences are contrasted to a case with the opposite result. The solution is a construction of an argumentation that will take many steps.

4.6 Complex Knowledge Sources

We introduce here a family of CBR applications that will be further discussed in Part IV. These are applications that deal with knowledge sources for the containers that demand a special body of methods to address. This category includes text, images, and speech. We illustrate text and images next.

4.6.1 Texts

When knowledge sources for a case-based reasoner are in textual form, the methods used are referred to as textual case-based reasoning. Methods and issues in textual CBR are discussed in Chap. 17, Textual CBR. In this section, we limit the discussion to an example of reasoning with cases where the sources are available as texts. Its major difference with other examples is that in the presentation of text no attributes are used.

Handling emails is one of the most time-consuming tasks computer users face these days. Emails are not pure text; they combine straightforward attributes like date and sender with short text in the subject field and unpredictable data in the body. Most email users have to deal with spam and with the limitations of spam filters. One of the main problems with spam filters is their rigidity. As spam emails change, so should the spam filters.

CBR is recommended for spam filtering classification because of how it compares to its alternatives. As described in Sect. 4.3.1 on classification, machine learning provides many highly accurate classifiers. The advantage of CBR is that since cases can be very different from each other, very distinct instances will be properly classified. Consider a collection of emails without attachments. Take each word and represent the emails by using all the words in all the emails as binary attributes. Thus, representing the emails means giving each attribute originated by a word a

value of either 1 if the word occurs in the email or 0 otherwise. Depending on the number of emails you start with, this representation can become cumbersome. Moreover, it is not very likely that every word is really significant for representing the emails. Hence, reducing the space of words is an important step. An example of a method to reduce the set of words is Information Gain, which is discussed in Chap. 22, Basic Formal Definitions and Methods. The resulting set of words is the vocabulary container. The represented emails are the case base container.

The similarity container requires a representation that can benefit from the fact that we do not have different weights for the attributes. This is the kind of problem where the relative significance of one word will vary in different emails. Furthermore, email spam is a domain-independent problem.

As mentioned before, the main problem with spam filtering is concept drift, that is, the fact that the nature of spam messages changes. In order to maintain a case base capable of sustaining classification accuracy, it is recommended that maintenance methods remove redundant or noisy cases and add exemplars of new spam forms. In general, a method to determine which cases to remove should be able to identify which emails each case helps to classify correctly and which it helps to misclassify. To add new cases, first add new positive and negative instances to the case base. Second, reassess the relevant features for the entire case base. This is a retraining step.

The implementation of such a system has to be done within an email management system. Each new arriving email becomes a new case, which is submitted to the CBR classifier for a decision on whether or not it is spam. It is necessary to be able to track classification accuracy to determine when an update is necessary. By adding a field to each new email, the classifier can assign or not the classification spam to this new field. As users revise the spam folder, they can modify the classification as a false positive and the spam messages that made it into the Inbox folder as false negatives, as applicable.

Example Highlights In this example, all words that occur in the emails are potentially good attributes for case representation. This is a use of CBR on a domain-independent application, where there is no knowledge available to determine the relative relevance of the attributes. Hence, all binary attributes are considered equally important. This example also uses evaluation methods.

4.6.2 Images

Images can occur in the query as well as in the solution. A simple situation occurs in pattern recognition: Which face from a catalogue is that of the person recorded in the video of a robbery? In medicine, problems are typically more complex. The fact that images contain knowledge is crucial, for instance, when:

- The query is an X-ray where one wants to know if it shows something pathological;
- The query is an X-ray and the expected solution is an image from an archive that is most similar to it.

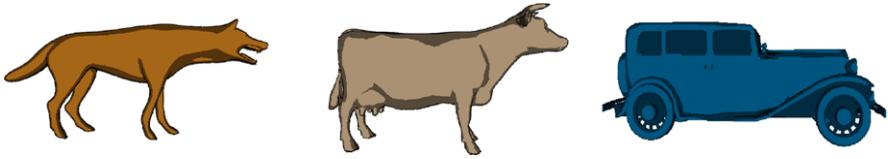


Fig. 4.12 Solution set



Fig. 4.13 Queries

As in texts, no attributes are used for the representation. An image is just a set of pixels. This implies that new techniques have to be involved, discussed in Chap. 18, Images. We consider images of animals or objects like cars, as in Fig. 4.12.

These objects constitute the solution set. A query is presented by a user who has seen such an object and draws a rough skeleton, as shown in Fig. 4.13.

Example Highlights From a query that is an image, retrieval has to adopt similarity measures that can understand the format in both the queries and the solution set. Special techniques are discussed in the Chap. 18, Images.

4.7 Chapter Summary

In this chapter examples of major problem areas as classification, diagnosis, prediction, call centres, etc. have been briefly introduced. We mainly discuss how example applications can be formulated in a form suitable for CBR. We do not present detailed solutions for each example. The examples show the variety of applications of CBR technology. The reader is advised to consult frequently these examples when reading other chapters of the book.

4.8 Background Information

The organisation and definition of tasks is inspired by Bloom's taxonomy (1956). The main categorizations we use are analytic and synthetic tasks. The rest of the chapter describes examples.

The example in medical diagnosis in Sect. 4.3.2.1 is based on the work of Marling and Whitehouse (2001). The example of diagnosis of technical faults is based on Althoff et al. (1989). The example of effort prediction is further described in Li et al. (2007).

Synthetic tasks include configuration, planning, and design. The bicycle configuration example is taken from the works by Kamp (1996, 2003). The case-based route planning in Sect. 4.4.2 is loosely based on the work of Haigh et al. (1997). The design example (Sect. 4.4.3) is based on the work in a system called Moneo (Taha 2006).

Section 4.5 includes examples of applications in call centres, e-commerce, knowledge management, and law. The call centre example in Sect. 4.5.1 is concerned with an automated conversation with a user. It has been developed as the Siemens Simatic Knowledge Manager (Bergmann 2002).

The e-commerce example in Sect. 4.5.2.1 of the house design is a standard exercise using the CBRWorks tool (2013). The knowledge management example in Sect. 4.5.3 is based on Weber et al. (2001). The example in Sect. 4.5.4 is based on Ashley (1988). The factor tree is inspired in the tree from Alevén and Ashley (1996).

Section 4.6 in complex knowledge sources starts with the example of email spam filtering in Sect. 4.6.1. It is based on ECUE, a case-based system that classifies email cases as spam or not (Delany 2006; Delany and Bridge 2006). Examples like the one in Sect. 4.6.2 on images are discussed in Chap. 18, Images. Much general information about sketches can be found in Igarashi et al. (1999).

4.9 Exercises

All the examples in this chapter can be the starting point for more advanced exercises and projects.

Exercise 1 Describe an application for configuring a library reference desk. Include case attributes.

Exercise 2 Create an application problem and domain that has not been discussed before.

Exercise 3 Describe a system containing a main and at least two auxiliary applications.

Exercise 4 Describe problem contexts for the following reasoning tasks: Design, Diagnosis, Prescription, and Planning.

Exercise 5 Create an example where cases could be described at two different levels of abstraction. Illustrate instances showing potential problems with both.

Exercise 6 Indicate adaptation rules for your solutions of the previous exercise.

Exercise 7 One can take any of the examples in this chapter and complete it to a working CBR system. This is not an easy task. It requires much time and knowledge from the subsequent chapters.

References

- Aleven V, Ashley KD (1996) How different is different? Arguing about the significance of similarities and differences. In: Smith I, Faltings B (eds) EWCBR-96: advances in case-based reasoning. Third European workshop, Lausanne, Switzerland, November 1996. Lecture notes in computer science (lecture notes in artificial intelligence), vol 1168. Springer, Berlin, p 1
- Althoff K-D, Faupel B, Kockskämper S et al (1989) Knowledge acquisition in the domain of CNC machining centres: the MOLTKE approach. In: Boose J, Gaines B, Ganascia J-G (eds) EKAW 1989: third European workshop on knowledge acquisition for knowledge-based systems, Paris, July 1989, p 180
- Ashley KD (1988) Modeling legal argument: reasoning with cases and hypotheticals. MIT Press, Cambridge
- Bergmann R (2002) Experience management: foundations, development methodology, and internet-based applications. Springer, New York
- Bloom BS (1956) Taxonomy of educational objectives: the classification of educational goals. David McKay, Ann Arbor
- CBRWorks (2013) What is CBR-Works. <http://cbr-works.net>. Accessed 22 Feb 2013
- Delany SJ (2006) Using case-based reasoning for spam filtering. Dissertation, Dublin Institute of Technology
- Delany SJ, Bridge DG (2006) Textual case-based reasoning for spam filtering: a comparison of feature-based and feature-free approaches. *Artif Intell Rev* 26(1–2):75–87
- Haigh KZ, Shewchuk JR, Veloso MM (1997) Exploiting domain geometry in analogical route planning. *J Exp Theor Artif Intell* 9:509–541
- Igarashi T, Matsuoka S, Tanaka H (1999) Teddy: a sketching interface for 3D freeform design. In: 26th annual conference on computer graphics and interactive techniques. ACM SIGGRAPH'99, Los Angeles, CA. ACM, New York, p 409
- Kamp G (1996) Using description logics for knowledge intensive case-based reasoning. In: Smith I, Faltings B (eds) EWCBR-96: advances in case-based reasoning. Third European workshop, Lausanne, Switzerland, November 1996. Lecture notes in computer science (lecture notes in artificial intelligence), vol 1168. Springer, Berlin, p 204
- Kamp G (2003) Fallbasierte Unterstützung von Experten im Bereich Service and Support. Dissertation, Hamburg University
- Li J, Ruhe G, Al-Emran A, Richter MM (2007) A flexible method for software effort estimation by analogy. *Empir Softw Eng* 12:65–106
- Marling C, Whitehouse P (2001) Case-based reasoning in the care of Alzheimer disease patients. In: Aha DW, Watson ID (eds) ICCBR 2001: case-based reasoning research and development. 4th international conference on case-based reasoning, Vancouver, BC, Canada, July/August 2001. Lecture notes in computer science (lecture notes in artificial intelligence), vol 2080. Springer, Berlin, p 702
- Taha D (2006) A case-based approach to computer aided architectural design. MONEO: an architectural assistant system. Dissertation, Alexandria University
- Weber RO, Aha DW, Becerra-Fernandez I (2001) Intelligent lessons learned systems. *Int J Expert Syst Res Appl*. 20(1):17–34

Part II

Core Methods

Part II presents the core problem areas in Case-Based Reasoning.

These areas adopt perspectives of both knowledge and process models. Chapter 5, Case Representations focuses on the vocabulary and case base containers. Chapters 6, 7, and 8 focus on the Retrieve step of the process model. Chapter 9 is about the Reuse step, while Chap. 10 includes Revise and Retain steps. Chapter 11 complements with development cycle and maintenance. These chapters present the main methods used in the respective problem areas. At the end of each chapter we discuss the question what should be considered when one has the choice between concepts and methods offered in the chapter.

Chapter 5, Case Representations is a good starting point for a CBR system. Case representations are fundamental in all tasks in the CBR methodology. Consequently, they will also be considered in the subsequent chapters in Part I, all the process steps operate on case representations. The chapter contextualizes the universe of representations and present formalisms that are ultimately based on attribute-value pairs.

Chapters 6 and 7 are about similarity. Similarity assessment represents a sub-step of the Retrieve process step. It is responsible for identifying cases of interest from the case base. Similarity assessment methods are stored in the similarity container. The basic idea about similarity is that if experience1 is more similar to the actual problem than experience2 then experience1 is more useful for the solution. Because usefulness depends on the user there is no universal method to define similarity. In fact, most of these two chapters present such methods. For this purpose different categories of similarity measures are introduced and motivated.

Chapter 8, Retrieval discusses methods to efficiently retrieve useful cases. This depends on the complexity of case representation and similarity assessment.

Chapter 9, Adaptation is concerned with the reuse of a retrieved experience. Because the previous situation is not exactly the same as the new one the retrieved solution may be very useful but has to undergo some change. This is called adaptation. The changes are performed in steps that are typically defined by rules.

Chapter 10, Revision and Learning describes how to evaluate a suggested solution. If the solution can be improved, then it is revised. In learning specific container elements as well as the whole system are improved.

Chapter 11, Development and Maintenance is the final chapter in Part II. It examines life cycle development steps and how these steps, in addition to other methods, can help maintain the quality of a CBR system. The chapter investigates developments of new systems and maintenance of existing ones from a systematic point of view.

Part II is comprehensive in that it provides a sound description of the CBR core problem areas. It is our understanding that these chapters' contents should be included in any regular course aiming to teach CBR. The comprehension of this entire Part II does not require any programming skills. Lecturers shall include programming requirements as they tailor exercises to specific student audiences.

Chapter 5

Case Representations

5.1 About This Chapter

As the first chapter in Part II, this chapter details and complements concepts from Part I. Therefore, a good understanding of Part I is expected. The chapter introduces representation methods, aiming to provide a sound understanding of the requirements, challenges, and solutions when designing, developing, implementing, adopting and deciding about case representations. It describes several case representation methods. It also serves as an introduction to more complex representation methods that are later explained in Part IV.

5.2 General Aspects

This chapter presents concepts and discusses issues related to how they are represented in cases and manipulated within the CBR methodology. It is organised as follows. It first presents a comprehensive categorization of case representation methods. Then, it presents these methods illustrated with examples, explaining potential problems and implementation aspects. In this chapter we want to present the most basic concepts in the way they are used throughout the book and their corresponding notations, which somewhat vary in literature.

5.2.1 Representation Layers

Humans share knowledge in various ways. They manage such communication mainly informally. However, to manipulate problems and solutions in an implementation, precision is needed. There are three layers for the representation of knowledge, as illustrated in Fig. 5.1.

At the cognitive layer the knowledge is informally presented by humans. Once knowledge is formalized for use in a CBR system, it is at the representation layer.

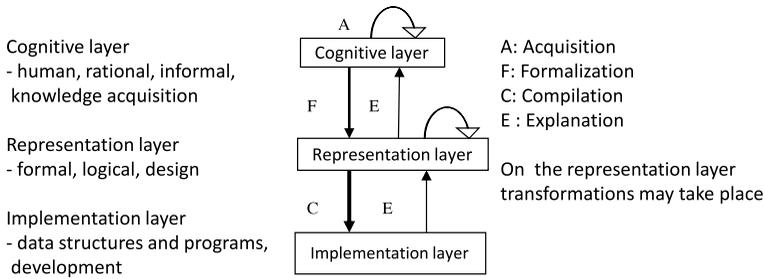


Fig. 5.1 Representation layers

When it is coded using data structures, it is at the implementation layer. These transformations go through knowledge acquisition, design and development stages. The CBR development life cycle is discussed in Chap. 11, Development and Maintenance.

Consider the example of someone who wants to buy a new car. Potential buyers think of cars informally. They may think they want to buy a car that is sporty, has low mileage, is loaded, and has a reasonable price. Such a cognitive layer description is also found in ads. Once this language is acquired and used in a computer system, it necessarily needs to be formalized. The car model, accessories, mileage, and price are replaced by vague adjectives used at the cognitive layer. In a CBR system, these can be, for example, values assigned to attributes that describe cases. At the implementation layer, each case is represented as a record (when databases are used) or an object (in object-oriented programming).

Knowledge at the representation layer may or may not be very easy to understand. The representations at the implementation layer are mainly addressed by computer scientists. However, all other professionals involved in the design and process need to communicate. When formal representations are not easy to understand, an alternative is the use of pseudocode. However pseudocode is not relevant within the system.

Case representations are also closely related to the decision of what kinds of information are to be represented in a CBR system. Their choice depends on the application context, the problem to be solved, the task to be performed, and the user class.

Often the knowledge to be represented is much more than just the problem and its solution. It is useful to include something about the context and how and where the solutions are used. Consider also paying attention to the expected knowledge level of the users. For instance, for a nonexpert user we may concentrate on simple functionalities and avoid technical descriptions of the products. In such situations one has to find a compromise between understandability of the represented concepts and the efficiency of computation. This is a known dilemma in computer science. These decisions are related to filling the vocabulary container; this is discussed in Chap. 3, Extended View. In Chap. 2, Basic CBR Elements, we showed that the same problem context can be described from different perspectives, depending on the intended application and the corresponding use. Examples we used were in repairing and buying, or selling a car.

The representation at the implementation layer has two further kinds of aspects:

- Formal mathematical ones, that is, the mathematical properties wanted;
- Aspects resulting from existing tools.

Representations at the representation layer can vary directly or indirectly:

- (a) Direct representations, using a representation language.
- (b) Representation via linking, often used in catalogues where only a small set of objects is represented directly. For the majority of objects there is a link to other directly represented objects and a way to modify an object in order to obtain many additional ones (for instance, to modify size). The latter approach is adaptation, discussed in Chap. 9, Adaptation.

Case representations can also be categorized based on whether or not their contents overlap.

Definition 5.1

- (i) A rule type case has no common concepts or variables in the problem and the solution description.
- (ii) A constraint type case has common concepts or variables in both descriptions.

Rule type cases occur frequently in diagnosis. The problem description contains the observations while the solution describes the therapy or repair and both have usually no terms in common. On the other hand, constraint type cases are typical for design problems, where the constraints for the objects under consideration are formulated and the solution describes the design found for these objects; the solutions usually refer to the same objects as the query. For instance, when building a house, one wants certain rooms, and the solution says how they are arranged.

The vocabulary determines also to some degree which data structures and which elements of the structures are used to represent notions and concepts.

The vocabulary has two distinct aspects:

- The syntactical form, for instance, the spelling;
- The meaning, i.e., the semantics.

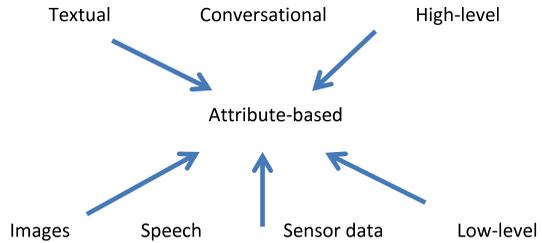
For any knowledge representation, it is relevant to find out which concepts with which semantics are chosen.

The challenges associated with defining the ideal representation exist on all three levels. In this chapter, we concentrate on the representation layer. Next, we introduce a variety of aspects that can come up when making decisions about representations.

5.2.1.1 The Representation Layer

In Chap. 2, Basic CBR Elements, we introduced case representations using attribute-value pairs. We also discussed how case bases can be represented, which refers to how cases are organised or structured.

Fig. 5.2 Different representations



There are other alternatives to attribute-value pairs. Overall, we can distinguish two main categories of representations that ultimately use some kind of attribute-value pairs. This chapter focuses on attribute-value representations. Other representations are discussed in Chap. 12, Advanced CBR Elements, and in Part IV.

In Chap. 2, Basic CBR Elements, we introduced representations where cases can be found:

- Attribute-value
- Text
- Images
- Speech
- Sensor data
- Conversational representation

These are not necessarily representation formalisms used to create representations. Rather, they are formats in which knowledge sources for cases are sometimes found. They are formats sometimes used to represent cases within CBR systems. Due to the fact that they are not typical representation formalisms, using them in CBR can be complex. Each of these forms has its own idiosyncrasies. Not surprisingly, we dedicate entire chapters to discuss them in Part IV.

The representations are all usually used in computer systems; they may even be used for case representation. Nevertheless, in CBR systems, it is often the case that they are ultimately represented through attribute-value pairs at the representation layer.

One of the reasons for this is the important role played by case representations in CBR. Case representations are compared by similarity measures, making the two intimately connected. For this reason, similarity is the topic of the two subsequent chapters, Chaps. 6 and 7.

Figure 5.2 shows the different case representations and how they may all be represented through attribute-value pairs. It also emphasizes the distinctions between those representations. Textual and conversational representations are at a high level. Conversational representations are distinguished by the fact that they can be represented in one step or incrementally. This is further discussed in Chap. 20, Conversational CBR. Images, speech, and sensor data are at a low-level; they are discussed in Chaps. 18 and 19.

Table 5.1 Effort involved with different representations

Effort involved	Attribute-based CBR	Textual CBR	Images
Reuse of knowledge	Medium	Low	Low
Initial modelling	High	High	High
Getting cases	Medium	Low	Low
Quality control	Low	High	High
Retrieval	Medium	High	High

As we will discuss in detail in Part IV, these representations, where cases can be found, can still be used to model cases in CBR systems. Table 5.1 gives a short overview of the effort involved when such representations are used. This is not meant to be used as a basis for deciding which representation to use, but rather what is involved when experiences are given in these forms.

These different levels of effort stem from the gap between the representation methods and the language a computer understands, namely, bits. This gap plays a role in development when cases at the representation layer are coded in the implementation layer. Attribute-based representations are closest to the computer; very often, there are standard compilers available. These representations are also useful for describing cases because they are amenable to most methods developed to execute the CBR process steps. For this reason, most other representations will ultimately be translated to attribute-based ones from different sources, as seen in Fig. 5.2.

Textual and conversational cases are easily understood by humans but structures need to be extracted for automatic processing. As with conversational representations, the naturalness of natural language has the counterpart of being distant from the bit representation level used by computers. Images, speech and sensor data are closer to the representation used by a computer, but may not be easily understood by the humans; they are at a lower level.

Attribute-based representations play an intermediate role. For automatic processing, the attributes are elementary symbols or signals and the structures have to be added. In summary, they are not directly understandable to a formal system or even to a computer. The reduction to attributes is discussed in the corresponding chapters, such as Chap. 19 for images and Chap. 19 for speech.

Chapter 2, Basic CBR Elements, also discussed representations of case bases. These are forms of case representations seen from the perspective of multiple cases at once.

The major representations for case bases that use attribute-value are:

- Flat attribute-value representation
- Object-oriented representation
- Trees and graphs
- Hierarchies and taxonomies
- Generalized (set-oriented) cases

Flat representations are sequential and linear case bases where each case is an attribute-value vector. All remaining are not flat, that is, cases are linked through

case features in various ways. However, they are still made with attribute values as atomic elements, which motivates the term “attribute-based” for all these methods.

In order to assess the quality of representations in general, we can use the following attributes:

- Representational adequacy, which asks “Is it possible to represent everything of interest”?
- Inferential adequacy, which asks “Can new information or knowledge be inferred”?
- Inferential efficiency, which asks “How easy (computationally) is it to infer new knowledge”?
- Acquisitional efficiency, which asks “How easy is it to formalize new information/knowledge”?
- Clear syntax and semantics, which asks “How easy is it to clarify what is allowed or not”?
- Naturalness, which asks “Are the representations easy to use and understandable”?

5.2.2 *Completeness and Efficiency*

Concepts and sets of concepts may have different properties that are of interest to applications. These properties are relevant for reasoning, and the terms in the section title, completeness and efficiency, are of major importance in this respect. The completeness of a set of concepts can be viewed by two criteria.

Principal completeness: All relevant properties can be formulated. If the concept set is incomplete in this sense then certain aspects or properties of interest cannot be represented in the system.

The second criterion considers the situation where it may happen that a certain relation between them is important for a decision. For instance, suppose for a person the income A and spending B is recorded for a period of time. In order to decide about the reliability of that person the important information may be the difference $C = A - B$. If this concept is missing the decision may get very complex. A related problem occurs when the original concepts are not operational but the value of C is computable. Then the missing knowledge has the consequence that the system is not directly aware of the relevance of C . For this reason it is advisable to add such additional concepts or attributes; we call them *virtual concepts or attributes*. Their addition can improve the efficiency of the system significantly and may even lead to the deletion of other concepts. We will discuss this in Chaps. 6 and 7 on similarity.

The efficiency considers two aspects too. The first one deals with the question of how time consuming it is to formulate the knowledge given in the specific representation formalism. This can heavily depend on the formalism.

The second question discusses the next steps in the process model and asks for the efficiency of computing the similarity measure and the retrieval. This will be discussed in Chaps. 6 and 7 on similarity and Chap. 8 on retrieval.

5.2.3 Flat Attribute-Value Representation

A simple and very useful representation uses attributes. Here we give a formal definition. In principle, attributes are fragments of predicate logic. They are much simpler and easier to handle and for many applications they are sufficient. This is explained in Chap. 22, Basic Formal Definitions and Methods.

Suppose we have two colours, red and blue, each represented by a predicate for an object a . If a is red, then the predicate $\text{Red}(a)$ is true. This representation becomes rather clumsy if we have many colours. In an attribute-value representation we can introduce an attribute $\text{Colour}(\cdot)$ that has a value set consisting of all colours.

For the basic definitions we suppose a universe of discourse U is given that can be an arbitrary set.

Definition 5.2

- (i) An attribute A has domain (also called type) $\text{dom}(A)$ which can be an arbitrary set.
- (ii) An attribute A assigns to each $a \in U$ an element $A(a) \in \text{dom}(A)$.
- (iii) Each object from U is represented as a vector (a_1, \dots, a_n) with $a_1 \in \text{dom}(A_1), \dots, a_n \in \text{dom}(A_n)$.

That means we know nothing about an object except its attribute-value.

Definition 5.3

- (i) A type T is a pair $(T_{\text{name}}, T_{\text{range}})$ where T_{name} is a label out of some type name space for referencing the type and T_{range} is a set which denotes the values that belong to the type. Further, $a \in T$ denotes a value of type T , which is short notation for T_{range} .
- (ii) The type space T is a set of types $\{T_1, \dots, T_n\}$ with disjoint T_{name} labels.
- (iii) While in standard programming languages types define allowed value ranges for variables, in case representations they define allowed values for attributes.

Often different domains contain only values of a specific type, for instance, integers, reals or strings. One distinguishes predefined standard types and user-defined types. Examples of standard types are:

- Numerical types such as integer or real,
- Symbol types that are defined by an enumeration of symbols,
- Textual types such as strings.

User-defined types are often subtypes of standard types. Some examples are:

- Weekdays as a subtype of symbols or strings.
- Small integers for the number of children in a class.
- Large integers for counting the inhabitants in a country.
- Special types for multimedia objects, such as images, videos or sounds, or URLs.

Set types are based on an arbitrary atomic type such as those listed before.

Table 5.2 Attribute types

Attribute types	Descriptions	Operations	Examples
Nominal	Enumerable values	$=, \neq$	Colour, hometown
Ordinal	Ordered-values	$=, \neq, \geq, \leq$	University grades
Interval	Interval of integers, reals	$=, \neq, \geq, \leq, +, -$	Range of airplanes
Real	Real numbers	$=, \neq, \geq, \leq, +, -, /, \cdot$	Distance, temperature

This is a very simple definition of attributes. In such a representation two elements that agree on all attributes cannot be distinguished; therefore, all objects are characterised by the vector of attribute values. In Chap. 2, Basic CBR Elements, an elementary and motivating example for describing cars was given. This is typical of diagnostic and classification problems.

Standard types contain little information (“string” is usually good only for names). Self-defined types can be problem-oriented. The choice of good types or attributes is often an important step towards the solution.

A typical selection of attribute types is shown in Table 5.2.

The attribute definitions can be generalized. The two important concepts are:

- Attributes with several values.
- Attributes with relations as values.

If attributes have several values, the value of the attribute is a set, then set-theoretic operations can be performed. Examples are:

- (a) Team members.
- (b) Meal ingredients.

Examples of relational attributes are:

- (a) Family relations: Brother, sister, mother, and so on; the relation holds between pairs of persons.
- (b) The part-of relation: A wheel is a part of a car. This is also discussed below.

The knowledge representation can be incomplete and two situations can occur:

- For some attribute a value is missing.
- A whole attribute is missing, i.e., the vocabulary container is incomplete.

For the representation of unknown values one can use special symbol “unknown” or “?” or a typed variable. The unknown value can have different interpretations. For instance, one can distinguish for “unknown” between “do not care”, “unfortunately not known but wanted”, and “waiting for the value”. In any case, one knows that some value is missing.

In addition, one can use default values. These are values that are taken normally, but not always. Somehow they are in between known and unknown. An example is the attribute “number_of_wheels” for a car, where the default value is 4.

If another value is given, then the default value is overwritten. Overwriting defaults always deserves special attention. For instance, it can generate the need for an explanation.

The incompleteness because of a missing attribute is more dangerous. Such an attribute does not say “hi, I am missing”, and this can result in a serious error. In science this is known as the problem of hidden variables. Suppose, for instance, an important attribute is missing for a machine failure. Then the resulting diagnosis can be misleading.

The opposite of missing values is redundant values. These are values that have no influence on the solution. The detection of redundancy is very useful; it is discussed below in the context of the influence relation. In the car example, other attributes like “number_of_valves” are redundant because the problem can be solved without knowing their values.

The set of attributes can be fixed or variable. If the set of attributes is variable then the vector representation (a_1, \dots, a_n) of the objects refers to the actual attributes and can therefore be of varying length. In diagnostic situations the number of attributes is usually variable.

Besides having missing attributes, one has an incomplete representation of the problem or solution, where attributes are not needed in principle but may be helpful. For example, their presence would simplify the procedures. These have been introduced as virtual attributes; see Chap. 6, Basic Similarity Topics.

The flat representations have different advantages and disadvantages. We give a compact summary.

Advantages:

- Simple and easily understood representation.
- Simple to store, for instance, in a database.
- Efficient retrieval possible; see Chap. 8, Retrieval.

Disadvantages:

- Representation of structural and relational information not directly possible.
- No information about sequences possible.

Recommendation:

- This representation is a first candidate if the case base is large and the cases are described by few features only.

5.2.4 Complex Representations in General

In this section, we will discuss several examples of complex representations. In order to relate different representations and functions to each other we need some structural principle that is shared by them and allows discovering commonalities as well as differences.

5.2.4.1 An Abstract View: The Local-Global Principle

In order to formulate a systematic approach we introduce a general structural representation principle. It is based on the view that (complex) objects are built up in

a systematic way starting from elementary (atomic) elements. The objects can be thought of as very general ones like machines, the human body, images, and so on. This systematic structure is called the local-global principle for complex object descriptions; it says: There are elementary (local or atomic) description elements; for simplicity, we assume that these are attributes.

Each object or concept A is (globally) described by some construction operator C from the local elements:

$$A = C(A_i | i \in I)$$

where I denotes some index set for the atomic elements A_i (i.e., the attributes).

The construction operator can generate a flat vector description or it can produce a very complex construction, for instance, describing a complex machine or building. This principle is applicable not only to objects occurring in a case description. It has a wide variety of other uses. Later on it will be used for describing similarity measures.

The local-global principle gives rise to two tasks:

The decomposition task: Break the object or concept down into atomic parts. Often, this task is needed because an object may be presented globally and the parts are initially unknown. Popular examples occur when one is faced with complex goals. Such goals have to be decomposed in order to be fulfilled.

The synthesis task: Compose an object or concept from simpler parts. This occurs when one wants to construct a complex object like a machine or a building. However, it plays also a role in getting a similarity measure, which will be discussed in Chap. 6, Basic Similarity Topics.

The principle allows us also to annotate parts if the constructed object is analysed further, for instance, with respect to importance or danger.

For the objects under investigation the local-global principle has very different realisations. The claim is not that the principle itself is very innovative; in fact, it is quite standard. The point is the unified use of the principle in order to allow a systematic treatment of the different techniques. This occurs frequently in everyday life and can suffice to solve problems.

5.2.4.2 Object-Oriented Representation

The simple attribute representation is flat and it cannot easily represent relations between attributes and hierarchies or taxonomies of attributes. For applications it is often useful to characterise objects or situations in a more complex way that has access to relations between attributes. An example is shown in Fig. 5.3.

Basically, attributes that belong together are integrated in a way that follows object description in programming languages.

The main principles are:

- An object describes a certain entity of the domain of interest by a finite set of properties where each atomic property is represented by an attribute.

Fig. 5.3 An object structure

Object class Hotel Name: String Location: String Single room: Room Double room: Room	Object instance: Beach Hotel Name: "Beach" Location: Las Palmas Single room: 22 Double room: 38
Sub-class Sport Hotel Sports: Symbol set	Object instance: Tennis Hotel Name: "Tennis Hotel" Location: Hamburg Single room: 43 Double Room: 25 Sports: Tennis, Volleyball

Fig. 5.4 Solution type

Object class Room Number: Integer Price: Real Room number: Integer

Fig. 5.5 Class hierarchy

<p>Class definition: Attributes and methods of the class Super classes Inheritance (in the class hierarchy) Kinds of attributes: typed attributes relational attributes many-valued attributes Example: Attribute colour_of_cartype colour. Attribute price_of_cartype integer. Class car_kind_of vehicle; attributes: colour_of_car, price_of_car.</p>

- An object class describes the structure of its objects using the attributes and their types.
- An object instance of a class assigned to its attributes.

On the left side we see the vocabulary used and on the right side the objects.

The object description can occur in the problem as well as in the solution. In the example the problem can ask for a certain hotel or room and the solution will offer a hotel or room as, in Fig. 5.4.

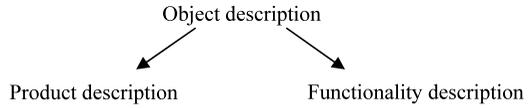
It is typical that such objects and classes are represented in a hierarchy. A simple way to represent a class hierarchy is in Fig. 5.5.

Simple object-oriented representations occur even for flat attribute representations. This takes place when one has a hierarchy of large sets of cases where the hierarchy distinguishes between possible values. As an example we consider a set of houses to rent where we have two parts in the city. Assume the first part has lower-class homes and the second one has upper-class homes. Then certain attributes coincide in each part and should not be repeated in each part. An object-oriented representation can realise this.

Advantages:

- The representation is closer to the application and easier to understand than flat representations.

Fig. 5.6 Sub-objects



- Certain aspects can be more easily be represented.
- More compact storage than for attribute-value representation.

Disadvantages:

- More complex computation for similarity and retrieval (see Chaps. 6, 7, and 8).
- Sequential orderings on objects cannot be formulated.

Recommended:

- For all problems where complex properties occur.

5.2.4.3 Relations Between Objects

Sub-objects can have various relations. We consider the situation where different object descriptions occur from different points of view. The two different views come from the producer and the customer. Then the object description has two sub-objects, as seen in Fig. 5.6. The product description is for the producer and the functionality description is for the customer.

In e-commerce the functionality is a demand from the customer and the product is an offer from the seller. The crucial relation between the two is “the product realises the functionality”. The example in Fig. 5.7 illustrates this.

The class Colour Printer has two instances, one called Wonder Printer and the other Best Printer. Both are specific objects that you can buy. Often, in a sales situation the functional description is an essential part of the query. The product solution needs to realise the functionality. As a consequence, the similarity measure has to judge the concept “realisation”. It is the task of the similarity measure

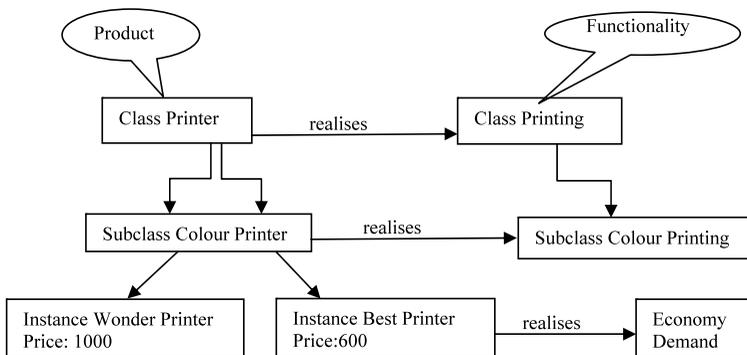


Fig. 5.7 Functionality realisation

to determine the most suitable case. This is discussed in Chap. 6, Basic Similarity Topics.

Functionality is not always a 0–1 property but can include degrees. Examples are standard requirements for technical devices, such as “producing heat”. Such degrees should also be reflected by the similarity measure.

5.2.4.4 Relations Between Objects and Attributes

To represent general knowledge about objects and cases, relations between objects and attributes are useful. There are many kinds of relations, which play different roles in applications. For example, there are spatial, temporal and logical relations. In general, the occurrences of such relations cannot be foreseen. However, there are some relations that frequently occur that have received particular attention. The major ones are taxonomic, compositional, and influence relations.

(1) Taxonomic relations: The is-a relation.

This relation expresses generalizations and specialisations. It is common to all object-oriented descriptions. A very simple interpretation of “A is-a B” is the subset interpretation: $A \subseteq B$, where A and B are sets. This relation is a transitive one. We show an example in Fig. 5.8.

This justifies the use of inheritance operations, for instance, in object-oriented programming languages. Inheritance means that an attribute or a function has only to be given for the more general object and not again for the more special object. In the example one can inherit the attribute “size”. The computation methods will differ from instance to instance.

Often, there are different hierarchies interleaved with each other and an object may occur in several hierarchies and inherit properties from several ancestors. This is called multiple inheritance. An example is given in Fig. 5.9.

Fig. 5.8 Taxonomy

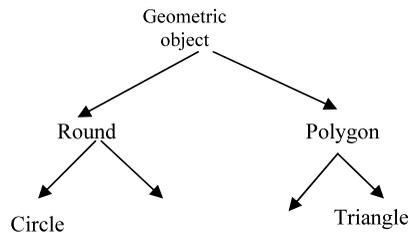


Fig. 5.9 Multiple inheritance



Fig. 5.10 Taxonomy problem

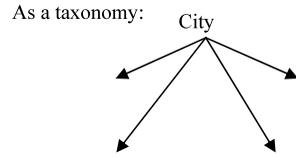
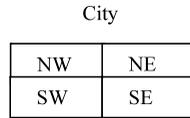


Fig. 5.11 Disturbing taxonomy

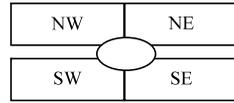
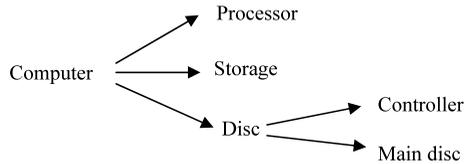


Fig. 5.12 Part-of relation



The relation “is-a” is, however, often used in extended ways, for instance, “mechanical engineering is-an engineering science”. It is not a priori clear to which sets the terms refer:

- The persons working in each science?
- The employees of the university departments?

An is-a hierarchy is represented by an acyclic graph but not necessarily by a tree. It can contain objects with several ancestors and there has not to be a top node. For instance, France is an European country and also an industrial nation.

Although taxonomies are very useful one has to be careful when dealing with them. Suppose one has a city divided from east to west by a highway and from north to east by a river, as in Fig. 5.10.

The city is, of course, divided further. In this taxonomy we can insert arbitrary useful attributes at the nodes and annotations at the edges. We can even implement an inheritance. But suppose now that we also want an area “downtown” that intersects all four parts, indicated by the circle in Fig. 5.11.

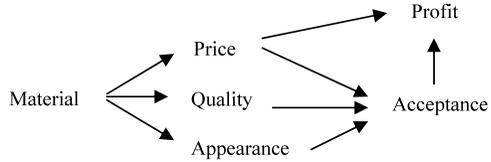
This leads to a complicated situation because downtown is not in the taxonomy. It can inherit some properties from all of the quarters. Therefore, it has to be questioned whether the taxonomy is the right representation in this situation.

(2) Compositional relations: *The part-of* relation.

This relation expresses the aggregation of objects from parts. This is a typical example of the local-global principle. Local aspects occur in construction and in repair while the functionality of the object needs to consider global aspects. An example is given in Fig. 5.12.

The parts are obtained by going from left to right. The part-of relation splits up because one can associate different meanings and intentions with it, for instance:

Fig. 5.13 Influence diagram



- *A* is a mandatory part of *B*.
- *A* is an optional part of *B*.
- *A* is an important part of *B*.
- *A* is a part of *B* which can be exchanged.
- *A* is an expensive part of *B*.
- *A* is a visible part of *B*.

These variations give rise to different uses. They are relevant not only for the part-of relation but also for dealing with more general situations concerning components.

The part-of relation is in general not transitive. In particular, a transitive view is not just simply false but often complete nonsense, as shown by:

“My leg is a part of me. I am a part of the university. Is my leg a part of the university”?

(3) The Influence Relation.

This relation is an important dependency relation. It is in the first place the binary relation “*A* influences *B*” where *A* and *B* are arbitrary objects, events, or properties. As a similarity, it has two forms, a relational one and a functional one:

- Influence as a binary relation.
- Influence as a function.

Because of the use for CBR we concentrate on influences between attributes, but they occur also between cases. Attributes with very low influence on the solution can mostly be omitted.

A net consisting of influence edges (if no feedback is involved, it is a directed acyclic graph) is called an influence net or an *influence diagram*.

The edges may be labelled with numbers describing the degree of the influence.

- This is a binary relation “*A* influences *B*”, where *A* and *B* are arbitrary objects or properties.
- A net consisting of influence edges (if no feedback is involved, it is a directed acyclic graph) is called an influence net or an influence diagram.
- The edges may be labelled with numbers describing the degree of the influence. An example is seen in Fig. 5.13.

Such influence diagrams are in general not complete. The usefulness of a diagram increases the more complete it is. The example shows the first arising difficulty because one may ask: Which has the greater influence, quality or price?

This leads to the more sophisticated relation “*A* influences *B* more than *C* does”.

Even more informative is to consider influence measures as real-valued functions:

$$\text{Infl: } U \times U \rightarrow [0, 1].$$

U contains the objects of interest. The user can determine a level of influence to consider an object as relevant. For this reason the influence relation is of particular interest for defining similarity measures.

A classical way to describe the influence of A on B is:

If small variations of A result in large variations of B , then the influence of A on B is high. This is the topic of sensitivity analysis, which can be complex.

A simple method uses the concept of predictive power (PredictivePower), which is introduced in Chap. 6, Basic Similarity Topics. This method is, however, not always applicable.

In order to control the influence in a net one has to consider the influence propagation by following a path in the underlying graph; however, it is often difficult to accumulate the influences when following the path.

Objects with a high influence on a decision or solution are usually called important. This plays a major role for similarity measures and we discuss this in Chap. 6, Basic Similarity Topics.

In a case description, attributes with no influences or an influence that is too small are redundant and should be omitted to simplify the system.

The is-a and the part-of relations give rise to tree structures and object-oriented representations. This is not the case for the influence relation. It can partially coincide with the part-of relation but often it will be orthogonal to both other relational structures.

A special case is when the attributes are random variables. Then the influence is expressed in terms of conditional probabilities. This allows propagating the influence by following the paths in the graph. The method is called Bayesian reasoning and will be discussed in Chap. 16, Probabilities.

5.2.4.5 Graph Representations

Trees and graphs are very useful for representing certain complex objects, in particular, because they allow visualisation and some efficient computation. Often, graphs are the underlying structure of many representations.

Some basic notions are:

Definition 5.4

- (i) An *undirected graph* is a pair (N, E) where N and E are finite sets.
 - N : Set of nodes;
 - E : Set of unordered pairs $(p, q) \in N \times N$ called edges.
- (ii) A *directed graph* is a pair (N, E) . N : A set of nodes.
 - E : Set of ordered pairs $(p, q) \in N \times N$ of edges.

- (iii) The size of a graph is usually just the number of edges; the order of a graph is the number of nodes.
- (iv) A path is a sequence of consecutive edges in the graph.
- (v) An attributed graph is a directed graph with additional marks on nodes and edges:
 - $\alpha: N \rightarrow V_N$, where V_N : node marks
 - $\beta: E \rightarrow V_E$, where V_E : edge marks
 Attributed graphs are often called semantic nets.
- (vi) A tree is an acyclic directed graph with exactly one special node that has no predecessors, called root. In the same way, a tree can be attributed.
- (vii) A forest is a set of trees.

The edges represent relations between the nodes. The marks on the edges and nodes of an attributed graph are also called annotations. Edges and nodes and their annotations are the local elements of the representations. The annotations can, for instance, describe the costs of going from one node to another. Such costs can be summed along a path, which gives rise to the problem of finding optimal paths.

The properties of the objects are indicated in the node attributes while the relations between them are in the edge attributes.

A special attributed graph is a net of objects. Each relational attribute becomes an edge annotation. The set of all attribute-value pairs becomes an edge annotation. Other typical examples of such objects are:

- Road maps.
- Networks.
- Descriptions of machines.

The machine descriptions can extend the part-of relation by allowing the representing of additional relations.

As an example we consider a pipeline problem. In a network there are nodes of different kinds given. There are two kinds of edges to connect nodes, directional and bidirectional. In addition, there are conditions on the possible edges that are not listed here.

The construction operators reflect more global properties by considering sub-graphs such as paths. This allows the formulating of queries to the global structure, for instance, for optimal connections between nodes.

A representation of the problem and the solution in terms of graphs is shown in Fig. 5.14, where the solution shows connections between the nodes. This problem occurs often when the edges denote wires for electricity.

A typical query for such a situation is, "How can I get efficiently from node x to node y ?" This is a highly complex combinatorial problem that in general prohibits the search for optimal solutions. Here, CBR can make use of experiences.

Graph representations can be extended by allowing the describing of additional options or restrictions. But this will lead to general object-oriented representations where the graph is only some part of the representation. This, however, occurs quite often.

Fig. 5.14 Graph representation

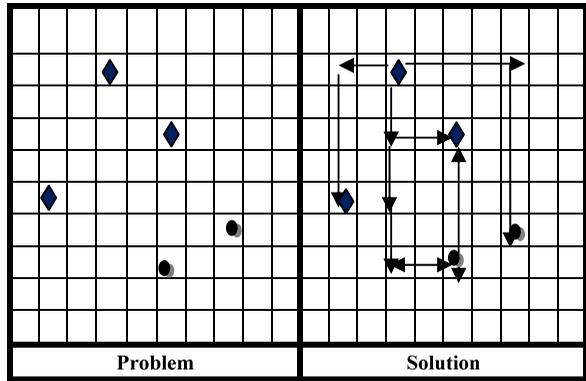
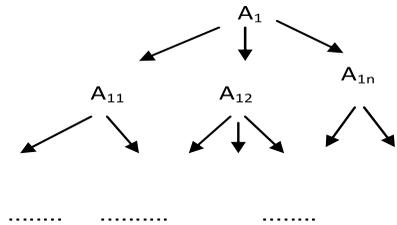


Fig. 5.15 Tree



5.2.4.6 Trees

In trees one can organise sets of objects as well as parts of an object, such as its attributes. The first possibility is useful for retrieval purposes; see Chaps. 8 and 14, on retrieval.

The second possibility allows a compact representation of several attribute-value vectors. This looks as in Fig. 5.15, where the attributes can be arbitrarily chosen.

One restriction of tree representations can be observed here: No relations between objects that are not connected (possible over a path of edges) can be represented. For instance, inner structures of nodes can only be discussed if the nodes are formally split up in the tree.

If an attribute A is a node label then for each element a in $\text{dom}(A)$ there is an edge below A . Hence each path from the top node to a leaf node describes an attribute-value vector and all such vectors are represented in this way.

Obtaining attribute-value tends to be costly. In the case of decision trees, an attribute may occur in different paths. In order to make a decision, it may not be always needed to use all the paths. Thus, one can shorten the tree in the sense that one stops going down whenever the decision is made. Therefore, one is interested in finding short trees; this is a topic in Machine Learning.

Trees are also commonly used to represent taxonomies. Going down the tree structure means going to more special objects or classes.

As an example we consider a city. A city is a complex object and a sub-object is a graph describing the roads and their connections.

The advantages are that the graph structure is very flexible; insertion and deletion of nodes and edges is possible unless additional constraints are violated.

The disadvantages are that the relations between objects other than the ones represented in the graph structure cannot be represented. In particular, inheritance is not possible. This is the price to pay for flexibility! The graph structure alone cannot store much knowledge.

The resulting message is that both trees and graphs allow some general object representations with few interrelations. They are still useful if the graph structure contains important information about the problem, for instance, temporal relations.

5.2.4.7 Sequences and Plans

Special graphs are directly ordered and, in particular, totally ordered.

Such a graph is of the form



Each edge has a start and an end node. Such graphs are used to represent temporarily ordered objects, in particular, actions. The most frequently ordered objects are plans and, in a more general setting, processes.

A plan uses a graph representations. The annotations use attributes or predicates.

The nodes describe states and the actions describe how a state node is transformed to the next node along an edge, in particular how the start node is transformed to the end node of the graph. A state can be something very general. In our case, state nodes are attribute-based or predicate descriptions. The intention is to change a state by some external action. Actions in a general form are therefore mappings:

$$\text{States} \rightarrow \text{states}.$$

That means an action transforms a state into a successor state by defining which of the previous facts are not true anymore and which new facts are true in the new state. An example of an action is the change of the colour of a car:

$$\text{Colour}(car) = red \rightarrow \text{Colour}(car) = blue.$$

We assume a set A of possible actions. For applying an action in a state, there are usually certain constraints given. In principle, the allowed actions are given by a plan graph. This graph is a partially ordered set of actions represented in the form

$$(\text{state } t_1, \text{action } a, \text{state } t_2).$$

The graph of all these allowed actions is called the plan graph. The problem of a planning case is of the form

$$P = (\langle \text{initial state} \rangle, \langle \text{plan graph} \rangle, \langle \text{goal state} \rangle).$$

A plan p is represented as a totally ordered subgraph of the plan graph, i.e., it is a path in the plan graph:

$$state_1 \rightarrow state_2 \rightarrow \dots \rightarrow state_n$$

A solution S to the planning problem is a plan p where $state_1$ is the initial state and $state_n$ is the goal state. For finding plans in real-world applications, one is confronted with two fundamental problems:

1. The Frame Problem:

How do we describe which features are not changed by the operators?

2. The Qualification Problem:

How do we avoid an infinite number of preconditions of operators?

These problems cannot be solved in general and one has to take care of them in the specific applications. This is a point at which cases can be useful.

The general way to describe actions is by using rules. Basically, rules represent the constraints for applying the actions. To proceed, we need first to describe formally what an action is. In general, actions are represented by operators that describe when some action can be performed. That means they are a form of IF-THEN expressions.

An operator is of the form $O(x_1, \dots, x_n)$ where x_i are variables; they are called the parameters of the operator. From the operators, one obtains actions by instantiating the variables. The colour examples can be represented by:

The operator $\text{ColourChange}(x_1, x_2)$.

Actions are $\text{Colour}(\text{red}, \text{blue})$ or $\text{ColourChange}(\text{white}, \text{blue})$.

A possible constraint is:

If $x_1 = \text{white}$ then $x_2 \neq \text{red}$, i.e., white cars can never be changed to red. This constraint restricts the rule to white objects and says nothing about other colours.

Applications of operators can be iterated and arranged in a sequence where the employed operators can be different.

Rules now describe the constraints under which an action can be performed. Formally a rule is of the form

$$\phi_1 \wedge \phi_2 \wedge \dots \wedge \phi_n \Rightarrow \text{Action.}$$

The first part of the rule description is called the preconditions; they represent the constraints. Rules are described in more detail in Chap. 9, Adaptation. In a CBR system the rules are in the adaptation container.

A common generalization is partial ordered plans. There the total ordering determining the sequence in which the actions have to be executed is replaced by a partial ordering. The main difference is that unordered actions can be executed in an arbitrary ordering. This is detailed in Chap. 9, Adaptation.

5.2.4.8 Processes

Very general objects occur when processes or workflows are under consideration. They have an underlying graph structure but the annotations are usually very complex. This is a very broad area and we make some remarks on the relation to CBR only.

One distinguishes between process models and processes or workflows.

A process model describes a set of processes; it allows us to define specific processes. Therefore, a process model may contain plan processes that have more than one method to choose from, for example:

- one may use Java or C++ for an implementation
- one may use the GUI_A or the GUI_B.

These methods are seen as alternatives and can occur in experiences. Therefore, they are stored as *cases*. For a specific project, proper methods need to be selected. This will be further discussed in Chaps. 6 and 7 on similarity.

5.2.4.9 Sets

Sets occur in different ways for representation issues. In the first way an attribute can have sets as values. This poses a problem to comparing sets using a similarity measure; this is discussed in Chap. 6, Basic Similarity Topics. A different situation arises when one considers sets of cases. Such sets are called generalized cases. There are two basic kinds:

- Sets in the classical sense.
- Vaguely defined sets, in particular, fuzzy sets.

Fuzzy sets are discussed in Chap. 15, Uncertainty. Generalized cases occur frequently in commerce, for instance, in catalogues. The reason is that the cases in the set have something in common that allows a compact representation by avoiding repetitions. That means such products are similar from the supplier's point of view. However, they may not be similar from the buyer's point of view because they may have quite different constraints and functional properties. The representation of generalized cases uses the following description elements:

The problem space $P = P_1 \times P_2 \times \dots \times P_n$ uses n problem attributes.

The solution space $S = S_1 \times S_2 \times \dots \times S_m$ uses m solution attributes.

A query is of the form $q = (q_1, \dots, q_n)$ using the problem attributes.

A case is of the form $c = ((p_1, \dots, p_n), (s_1, \dots, s_m))$.

For example, consider the design for an electronic synthesis tool for a discrete cosine transformation where the variables are: f : frequency; a : area; w : width; s : subword.

The generalized case is defined by constraints that restrict the possibilities for individual cases in the set. These constraints are formulated here in terms of inequalities and equalities as shown in Fig. 5.16.

$$f \leq \begin{cases} -0.66 \cdot w + 115 & \text{if } s = 1 \\ -1.94 \cdot w + 118 & \text{if } s = 2 \\ -1.75 \cdot w + 88 & \text{if } s = 4 \\ -0.96 \cdot w + 54 & \text{if } s = 8 \\ -2.76 \cdot w + 57 & \text{if } s = no \end{cases} \quad a = \begin{cases} 1081 \cdot w^2 + 2885 \cdot w + 10064 & \text{if } s = 1 \\ 692 \cdot w^2 + 2436 \cdot w + 4367 & \text{if } s = 2 \\ 532 \cdot w^2 + 1676 \cdot w + 2794 & \text{if } s = 4 \\ 416 \cdot w^2 + 1594 \cdot w + 2413 & \text{if } s = 8 \\ 194 \cdot w^2 + 2076 \cdot w + 278 & \text{if } s = no \end{cases}$$

Fig. 5.16 Generalized case

In this way one gets a very compact and a convenient representation of many individual cases. This is typical of technical catalogues that describe a set of products, which may be understandable only by an expert. The fact that they are represented in compact form does not mean they have the same properties from the user's point of view. That is, from the user's point of view they may not be similar.

Vaguely defined sets occur frequently in business. A typical example is customer classes. On the one hand, only some prototypical persons belong to a certain customer class, but only to some degree. For different classes, different similarity measures are needed because the classes have different preferences.

Therefore, it is natural to split the case bases into subbases. For example, people who buy famous brand products, generic products, and ecological products. These classes are vague and not disjoint. Because there is no class definition available, one needs a learning algorithm. In Chap. 10, Evaluation, Revision, and Learning, we sketch clustering algorithms in this context.

5.2.4.10 Representations via Links and References

Links provide access to documents. The purpose of these indirect representations is twofold:

- To save space.
- To give a more compact, systematic, and easier-to-understand representation.

From the user's point of view, one wants advice on how to use this representation form, and the seller's task is to define and implement the allowed links. That means the user has an expectation of what the link has to offer and the seller has to know these expectations and to select the link properly.

Formally, the links are URLs represented as strings. They represent, however, a data structure in its own right and used in a very special way.

5.2.4.11 The Knowledge Contained in the Case Representation

There are two kinds of knowledge. One kind is the explicitly represented knowledge. This is what belongs partially to the vocabulary container. It includes the annotations of edges and nodes of a graph.

The other kind of knowledge is in the structures of the representations. These structures are usually equipped with some methods as inheritance. Another example

within this same kind is the knowledge leading to directed or undirected edges in the graph.

Overlooking such knowledge will complicate the necessary computations with respect to similarity, retrieval, storage space and understanding.

5.2.4.12 Which Case Representation Should I Use?

This is one of the very first decisions a system developer is confronted with. The answer depends of course to a great extent on the application and the available data for the cases. If the cases are contained in a database then one should use their format and download them.

Sometimes it is possible to automatically transform one representation into another one. However, when the application is described in a very informal way there may be a lot of freedom for choosing the formal representation.

As a first step the situation can always be analysed from the viewpoint of the local-global principle and be represented in a graph. Therefore, the first question raised is, “What are the atomic elements of the objects under consideration”?

They may be, for instance:

- Attributes,
- Texts or text documents,
- Images,
- Links.

This allows us to distinguish between formal and informal entries. Often, objects are given in a flat representation. That does not mean we must use them as they are given. For instance, one may be given the parts of a structure in an alphabetical ordering as the members of a team. For further processing it can be useful to consult other information sources for defining a more complex structure, for instance, a team hierarchy or categories.

As a second step one has to analyse the connections between the elements. That means one has to find out the parts of the construction process for a global structure, describing the whole team, for instance. This has an influence on the chosen similarity measure.

The last step is considering retrieval. The speed of retrieval depends on the size of the case base and the chosen representation formalism. This is discussed in Chaps. 8 and 14 on retrieval.

5.3 Tools

Most CBR tools have a function for downloading cases from a database, comma separated files, and so on.

There are many tools for converting represented knowledge. Tools for creating concept maps are offered by the Institute for Human and Machine Cognition. Their tools are available at <http://cmap.ihmc.us/download/>.

5.4 Chapter Summary

In this chapter different attribute-based representation forms for cases have been investigated. A distinction was made between explicitly represented objects and those represented via links or references. The simplest form introduced is the flat attribute-value representation, including a formal definition of attributes.

In order to deal with more complex representations a general construction principle was introduced: the local-global principle. This principle allows describing all further object representations shown. They include object-oriented, hierarchical, taxonomies, and graph-oriented representations. Trees are used for compact representations of attribute-value vectors. Graph structures often underlie more general object-oriented representations.

An application of graphs is to represent plans, process models, and processes. A compact representation of cases can also be achieved by generalized cases that are sets of cases.

5.5 Background Information

Knowledge representation was systematically investigated in Artificial Intelligence in the area of Knowledge-Based Systems. These representations are based on mathematical logic, mainly predicate logic or one of its variations or fragments. This uses mainly declarative languages although knowledge is also contained in algorithms.

An essential difference to CBR is that classical knowledge-based systems have an explicit deductive system for reasoning, as opposed to CBR systems, where reasoning takes place using similarity.

An overview of graph representations is given in Riesen and Bunke (2009). Object orientation for CBR is discussed in Bergmann et al. (1996). There are also many textbooks on object-oriented programming languages, for instance Wang (2008).

Solution algorithms for finding a short path in a network are, for example, incorporated into the Google search engine. They rely on the discovery of the “small world” phenomenon (Kleinert 2003). Small world here means that even in a huge network there often exist very short paths between nodes, and the problem is to find them.

An early reference for planning in AI is Tate (1977). Hierarchical task networks are discussed in Ghallab et al. (2004).

5.6 Exercises

Exercise 1 Suppose you have an electronic shop that buys and sells laptops. Describe your laptop from the viewpoint of:

A seller

A user who is a computer expert

A user who is a layman

Exercise 2

- (a) Describe your laptop in a part-of representation. Distinguish between the different kinds of this relation mentioned in the text.
- (b) Construct two influence diagrams, one for the price and one ease for the use.

Exercise 3 Extract a partially ordered plan from your curriculum of classes to be taken.

References

- Bergmann R, Wilke W, Vollrath I, Wess S (1996) Integrating general knowledge with object-oriented case representation and reasoning. In: Burkhard H-D, Lenz M (eds) Case-based reasoning-system development and evaluation. 4th German workshop, Humboldt University, Berlin, 1996
- Ghallab M, Nau D, Traverso P (2004) Automated planning: theory and practice. Elsevier science and technology books. Morgan Kaufmann, San Francisco
- Kleinert J (2003) Bursty and hierarchical structure in streams. *Data Min Knowl Discov* 7(4):373–397
- Riesen K, Bunke H (2009) Graph classification by means of Lipschitz embedding. *IEEE Trans Syst Man Cybern, Part B, Cybern* 39(6):1472–1483
- Tate A (1977) Generating project networks. In: *IJCAI 1977: 5th international joint conference on artificial intelligence*, Cambridge, MA, 1977, vol 2. Morgan Kaufmann, San Francisco, p 888.
- Wang PS (2008) Java with object-oriented and generic programming. sofpower.com. Stow, OH

Chapter 6

Basic Similarity Topics

6.1 About This Chapter

This chapter assumes understanding of Part I and Chap. 5, Case Representations. There are two chapters on similarity in Part II and they use concepts from Chap. 5. This first chapter is intended to give the reader who wants to understand or to design a CBR system a first advice on which measure is suitable for which task. For this purpose, it contains the basic material necessary for understanding and building simple similarity measures.

Similarity is a basic concept in CBR. The principle is independent of the representation method. Occasionally we will mention similarity when making remarks on topics such as fuzzy sets, random variables and texts that are investigated in later chapters.

6.2 General Aspects

Similarity is a common term in everyday language. Since the Tower of Babel there has rarely been such a divergence of views as on the interpretation of this term. In a popular view, two objects are considered as similar if they look or sound similar. This has always played a role in subjective interpretations, as is the case with images or music. This term similarity is not nearly as clearly defined as the term equal. Hence, it is accepted that similarity is a subjective concept, suited for approximate rather than exact reasoning.

The subjective view and its many different realisations are discussed in cognitive psychology. Such considerations played an important role in cognitive science under the term same-different judgments.

A more refined use of similarity is when one says that two objects are similar in a certain respect, for instance, they are similar in size or colour. This limits the term to the dimension in focus. A consequence is that objects with a similar size may not look similar in a naïve sense anymore. In addition, the remaining aspects

are no longer used as the basis for assessing similarity. This applies, in particular, in problem solving with CBR. The so-called CBR paradigm can be formulated as:

Two problems are similar if they have similar solutions.

This is not really a paradigm or a general assumption; it is rather a demand for defining a proper similarity concept. This means one has to find aspects of interest in each problem situation in order to be able to identify an adequate definition of similarity. In this chapter the notion of similarity with respect to the objects of interest is discussed in a systematic way that is not restricted to a naïve view.

The measures operate on case representations and therefore depend on them. As a general construction principle we make use of the local-global principle (introduced in Chap. 5, Case Representations). We focus on how to find a measure that is suitable for such a given task.

6.3 Similarity and Case Representations

Similarity concepts are defined for the comparison of objects such as cases; more precisely, they operate on representations of the objects.

In CBR, similarity is the essential function used for retrieval. For this reason similarities link the concept representations and retrieval:

Case Representations — Similarity Measures — Retrieval

For constructing similarity measures we refer to Chap. 5, Case Representations, and follow it step by step. This will be continued in Chap. 8, Retrieval.

In the CBR process, cases appear as either query problems or complete cases. As previously seen, there is a great variety of case representations, and they are reflected in a corresponding wide range of similarity concepts. In this chapter we consider attribute-value representations only.

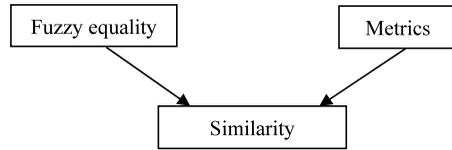
As mentioned, similarity is very general and it applies to many other forms of representations. Other representations that we will consider later on are:

- Texts,
- Images,
- Fuzzy sets,
- Random variables.

A minimal but essential requirement for defining a similarity measure is that the data structures of both case representation and measure be compatible. When they differ too much, one has to provide either some kind of dictionary or an intermediate language between them two to make them compatible.

Simple representations need simple similarity concepts only, while complex representations require more effort. However, case representations take precedence

Fig. 6.1 Influence factors of similarity



given their role in the CBR process; consequently, decisions about similarity measures should follow the representation and adjust to be compatible with them. Because we here consider attribute-based representations the similarity measures will be attribute-based too.

6.3.1 *Mathematical Models of Similarity*

Because similarity is central to CBR, it should also be based on a rigorous foundation. In this section, we will present principal aspects, basic definitions and elementary examples. The models for similarity are of mathematical character and computation-oriented so that they can be easily implemented. A deeper analysis is given in Chap. 13, *Advanced Similarity Topics*, in Part III.

6.3.2 *Meaning of Similarity*

In order to understand the scope of mathematical models that can be used for similarity, it is helpful to think of two influencing factors of the similarity concept (later on we show other influences), fuzzy sets and metrics. Fuzzy sets are discussed in Chap. 15, *Uncertainty*.

Earlier we mentioned that similarity is not an exact concept. We also explained how similarity between objects can be limited to one or more dimensions as when observing the similarity between two objects with respect to given selected dimensions.

Fuzzy set theory provides the theoretical background to model inexact expressions. It is possible to express the concept of “almost equal” through a fuzzy equality. This captures an important aspect of similarity.

Metrics, or, more generally, distance functions, play a central role in mathematics. They are used whenever approximations (rather than exact solutions) are involved, making them suitable for modelling similarity. This will be investigated in detail in this book. It should be remarked that there are many metrics, depending on the intended approximation. In this respect there is no difference between them and similarity concepts, where also a rich variety exists. We show the relation in Fig. 6.1.

From the two influence factors, the similarity concept has inherited not only properties of intuitive character but also certain precise notions, axioms, and techniques for applications.

Table 6.1 Possible axioms

	Fuzzy equality	Metric
Reflexivity	$E(x, x) = 1$	$d(x, x) = 0$
Symmetry	$E(x, y) = E(y, x)$	$d(x, y) = d(y, x)$
Transitivity	$E(x, y) \leq \sup\{z \min(E(x, z), E(z, x))\}$?
Triangle inequality	?	$d(x, y) \leq d(x, z) + d(z, y)$
Substitution axiom	$(s(a) = t(a) \wedge a = b) \Rightarrow (s(a) = t(b))$?

Similarity and distance look at the same objects from different points of view: similarity says how close objects are and distances say how far apart they are. There are different methods to implement these views to find close and distant objects. Some methods are more adequate for finding something close while others are more adequate for finding something far away.

The fuzzy view is recommended because we do not deal with classical yes-or-no answers but rather ones that have a vague character. We give a short summary of axiomatic properties in Table 6.1.

The mark “?” means that it is not clear what an adequate analogue is.

Often, transitivity is seen as a counterpart of the triangle inequality. But these two concepts are not logically equivalent. In fact, it is not even clear what equivalence should mean here. Up to now we have not shown general methods for comparing fuzzy sets and distances. These and other possible axioms have been questioned and will be discussed below.

However, the two concepts fuzzy sets and metrics have been formulated in many different ways with many different intentions. For similarity we will discuss this in a systematic way. In particular, the goal for using them will play a central role and will lead to utility as a common root for the different intentions.

An immediate question is, why are there so many different similarity relations and measures? There is only one concept for equality and this is universal; so, why is there not a universal similarity concept?

There are two main reasons:

- (a) Similarity depends on the context to a much higher degree than equality: Similar with respect to what? For instance, when are two cars similar? We can consider the shape, the size, the number of doors and many other properties.
- (b) Equality is crisp, “yes or no”, while similarity is in degrees. One cannot say that two objects are more equal than two other ones, while this is central to similarity.

6.3.2.1 Relations Versus Functions

There are two mathematical ways to represent similarity:

- (a) As a relation.
- (b) As a function.

The simplest relational way is to regard similarity as a binary relation. For simplicity, we assume first that all objects we compare are taken from the same underlying set, which is not further specified. Later on, this will be generalized. We start, as mentioned, by representing similarity in the two types of mathematical models.

6.3.2.2 Similarity as a Relation

The use of relations for describing similarity is not restricted to any representation method because it does not refer to one.

A slightly more sophisticated relational form is qualitative, and it just compares different similarities. In the comparison one says, for instance, that two objects are more similar to each other than two other objects.

There are three kinds of relational models:

- (i) A binary similarity predicate:

$$\text{SIM}(x, y) \Leftrightarrow \text{“}x \text{ and } y \text{ are similar”}$$

- (ii) A binary dissimilarity predicate:

$$\text{DISSIM}(x, y) \Leftrightarrow \text{“}x \text{ and } y \text{ are dissimilar”}$$

- (iii) Similarity as a partial order relation:

$$R(x, y, z) \Leftrightarrow \text{“}x \text{ is at least as similar to } y \text{ as } x \text{ to } z\text{”}.$$

Obviously, the similarity and the dissimilarity predicates formulated in this way are negations of each other. Viewing similarity as a binary predicate may be too simple and hinder the ability to grasp the intuition of the intended modelling of the approximation. On the other hand, if similarity is the basis of a binary decision, the relational view is essential, but we would rather use the formulation “sufficiently similar”.

The predicate in (iii) is more powerful in expressiveness, because different degrees of similarity can be distinguished. This is closely related to the quantitative view but formally different.

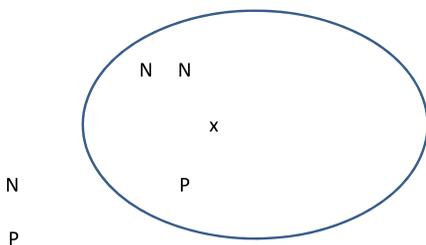
A generalization of (iii) is

- (iv) $S(x, y, u, v) \Leftrightarrow \text{“}x \text{ is at least as similar to } y \text{ as } u \text{ is to } v\text{”}.$

This allows us to define R (i.e., a partial order relation) as $R(x, y, z) :\Leftrightarrow S(x, y, x, z)$. It generates a partial ordering with respect to a given object x . The ranking shows which other objects are more or less close to x .

The concept of similarity is closely connected to the neighbour concept. Naively speaking, two objects are neighbours if they are regarded as similar.

Hence the most similar object y to a given object x will be called a nearest neighbour; the nearest neighbours are the set of most similar elements, the top elements of the ranking.

Fig. 6.2 Votes of neighbours

These relations allow making this more precise by introducing formally the nearest neighbour concept.

Definition 6.1 For some fixed x , each y that satisfies $R(x, y, z)$ for all z is called a nearest neighbour of x . Notation: $NN(x, y)$.

That means that NN is a relation and the nearest neighbour is not necessarily uniquely defined; there may be several such (equally similar) elements. This notion is extended to the first k -nearest neighbours.

Notation: The first k -nearest neighbours to x are listed in a sequence according to their neighbourhood ranking:

$$NN_k(x) = (z_1, \dots, z_k).$$

In applications, where effective similarity measures are built, often the nearest neighbour relation is the most important one. The reason is that *closer* has in general the meaning of *better*; the nearest neighbour is the best. We will further discuss this in Chap. 13, Advanced Similarity Topics. Here we keep in mind that *more similar* is always interpreted as *more useful*.

The computation of the nearest neighbour is known as the retrieval problem, discussed in Chap. 2, Basic CBR Elements, and it is a large problem area on its own; see Chaps. 8 and 15, on retrieval. Nearest neighbour computations are central to CBR.

There are different ways to make use of the k -nearest neighbours. Should one consider only the nearest neighbour e or also the following neighbours? And what kind of influence should they have?

One way is that the k -nearest neighbours vote. Suppose we have a classification problem with two classes P and N and the case base contains objects that are already classified. If a new not yet classified object x occurs we can make use of the case base in two ways, as shown in Fig. 6.2.

We can classify x as its nearest neighbour and use 1-NN. This classifies x as this would classify X as P . We can also use the k -nearest neighbours as voters; for instance, we use 3-NN. This would classify x as N .

Another way is to introduce weights for the neighbours if we feel that the voters are of different importance, for instance:

The vote of the i -nearest neighbour counts $k - (i - 1)$.

This says that closer neighbours have a higher influence on the solution when we take over the solution from the neighbours. The voting principle is very basic to most kinds of similarity computations, in particular, for defining similarity measures. It is intuitively clear that certain aspects contribute more to a similarity than others do. What aspects these are depends on the underlying intention.

Whenever a basic relation type is introduced, one asks for general properties that can be formulated in the form of axioms. There are several possible properties (“axioms”) that the relation S can benefit from; the most common are that for all x , y , z , u , v , s , and t the following hold:

(1) Reflexivity:

$$S(x, x, u, v)$$

Every object is at least as similar to itself as two other arbitrary objects could be to each other.

(2) Symmetry:

$$S(x, y, u, v) \Leftrightarrow S(y, x, u, v) \Leftrightarrow S(x, y, v, u)$$

This is how it is expressed formally. Symmetry basically means that x is as similar to y as y is to x .

(3) Transitivity:

$$S(x, y, u, v) \wedge S(u, v, s, t) \Rightarrow S(x, y, s, t)$$

Transitivity refers to a binary relation. In particular, similarity remains valid if we obtain it by arbitrary long chains. This is inherited from equality but neglects the fact that similarity is an approximate notion.

In a first view, these axioms are intuitively plausible and they are mostly inherited from the influence factors. However, later on in Chap. 13, Advanced Similarity Topics, we will get back to this matter and will question such axioms.

If one puts the focus on a fixed object x (usually the current problem) we have additional useful relations induced — \geq_x , $>_x$, and \sim_x — defined by:

- $y \geq_x z \Leftrightarrow R(x, y, z)$ “ y is at least as similar to x as to z ”;
- $y >_x z \Leftrightarrow (y >_x z) \wedge \neg(z \geq_x y)$ “ y is more similar to x than to z ” (strict part of \geq_x);
- $y \sim_x z \Leftrightarrow (y \geq_x z) \wedge (z \geq_x y)$ “ y and z are indistinguishable”.

All these relations are purely definitorial, and no way of computation has been discussed till now. Therefore, they are independent of the representation of the objects under consideration. This will be quite different for functional models where computations are essential and therefore the representation methods enter the stage.

The intuitive idea of “ x is more similar to z than y ” is that x is preferred over y with respect to z . This is a preference relation in utility theory and this view will be used for defining the semantics of similarity in Chap. 13, Advanced Similarity Topics. It also means that the nearest neighbour is the most preferred or the object with the most utility; this explains the popularity of the nearest neighbour concept.

6.3.2.3 Similarity as a Function

A functional way to make similarity quantitative is by expressing numerically how similar two objects are. This is more detailed and more expressive but also more difficult to obtain and prone to errors.

First, we start again with some general definitions that are independent of the representation of the objects to be compared.

For each case base there is on an abstract level the problem of finding the nearest neighbour. The interesting situation is the one in which we have many cases to choose from. For this reason, similarity measures are used that make the arguments more precise.

Similarity functions (“measures”) are used to refine the relational approaches quantitatively by assigning a degree of similarity to two objects.

Definition 6.2 A similarity measure for a problem space P is a function

$$\text{sim} : P \times P \rightarrow [0, 1].$$

This occurs, for instance, when new and old problem situations are considered. Higher values of sim denote higher similarities, and hence more usefulness. The maximal similarity value is 1, the highest utility; and the minimum is 0, which means not similar at all or the least utility. Now we want the best case, denoted as the nearest neighbour.

The maximal and minimal values can be formulated on an ordinal scale, but here we discuss them on a quantitative level that extends the ordinal one.

Definition 6.3 For a given problem P , a nearest neighbour is a problem P' that has maximal similarity among the problems in P .

For a given similarity measure the retrieval of the nearest neighbour is a problem on its own that is treated in Chap. 8, Retrieval.

Similarity should be computed fast. For this reason, index structures that support fast searching are used. Such index structures provide a path for finding the desired case. An example of a very elementary form is that of searching using an addressee email in an email management system. The address is structured in such a way that it encodes a path to the addressee.

A more complex example is that of searching for product descriptions, as discussed in Chap. 3, Extended View. In some sense this is made difficult because of the need for comparing objects that are expressed in different terms. This goes beyond simple syntactic means for defining the measure. We will discuss the retrieval problem extensively in Chaps. 8 and 14 on retrieval.

The notion of similarity can be extended to relate elements of different sets to each other:

$$\text{sim} : U \times V \rightarrow [0, 1].$$



Fig. 6.3 Degrees of dissimilarity

This occurs for instance when one compares an intended functionality of a machine with the product description. Another example is when one compares symptoms and therapies in medicine. Even for these extensions, similarity functions can be considered as fuzzy sets of ordered pairs. Here that again means that similarity is in degrees and not a binary similar or not similar choice.

The extended view as discussed in Chap. 3, Extended View, applies to many relations. These are far from the naive view of “looking similar”. Below we will see examples.

From fuzzy equality, similarity functions have inherited four possible axioms for all x, y, z :

- (1) $\text{sim}(x, x) = 1$ (reflexivity)
- (2) $\text{sim}(x, x) = \text{sim}(y, y)$ (constancy of self-similarity)
- (3) $\text{sim}(x, y) = \text{sim}(y, x)$ (symmetry)
- (4) $\text{sim}(x, y) \geq \sup(z\{\inf(\text{sim}(x, z), \text{sim}(z, x))\})$ (transitivity)

These axioms will be disregarded or at least relaxed when dealing with certain applications; this will be discussed later on. The requirement that similarity functions be bounded to $[0, 1]$ is a kind of restriction too. It may lead to difficulties and may also be given up.

Besides the ordinal information, similarity measures have also a quantitative statement about the degree of similarity. In particular, often objects with a very small similarity score to each other are called dissimilar, as illustrated in Fig. 6.3.

In Fig. 6.3 y_1 and y_2 have very high similarity to x , which would justify our calling them “almost equal to x ”, while y_3 and y_4 have a very small similarity to x and we would call them “quite dissimilar to x ”. This could not be expressed in a purely ordinal way unless there is an extra predicate *dissimilar*. But then the problem arises when relating similarity to dissimilarity.

By its very nature, similarity contains an essential element of uncertainty. On the other hand, if similarity is involved in some process, for example, for reaching a decision, one often encounters a “yes or no” problem.

Nevertheless, in practice, the goal of retrieval will ultimately be to determine whether or not a candidate case should be recommended as the suggested solution to the query problem. This is ultimately a binary decision, that is, yes or no. The process of taking an uncertain value and converting it into a crisp one is called defuzzification (see Chap. 15, Uncertainty). Though multiple methods for defuzzification can be used, we now present a simple one based on Rough Set Theory, also discussed in Chap. 15.

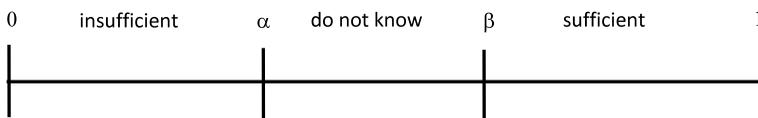


Fig. 6.4 Uncertainty area

The defuzzification process uses thresholds to determine intervals as shown in Fig. 6.4. Three intervals are defined to cover the concepts of *insufficient*, *do not know*, and *sufficient*.

The underlying notion is that, under the uncertain terms that the similarity represents, it is at least reasonable to identify intervals where the resulting similarity is sufficient and where it is insufficient. In order to cover for the ignorance of the exact crisp point where it is neither *sufficient* nor *insufficient*, a *do not know* interval is defined.

The result of the defuzzification process is, “If the similarity is in the *insufficient* area, then the solution is not accepted; if it is in the *sufficient* area, then it is accepted”. The *do not know* area is the uncertainty area and one needs additional criteria for a decision.

This view can be extended by considering similarity measures with negative values. Consider now:

$$\text{sim} : U \times U \rightarrow [-1, 1].$$

In theory, this can be achieved by encoding $[-1, 1]$ into $[0, 1]$, but it is sometimes easier to consider the negative numbers too. This can directly be used to give a penalty if the similarity is negative.

The dual concept for similarity measure is the distance measure (or function) that is supposed to represent dissimilarity.

6.3.2.4 Distances

Distances are a kind of proxy to similarities. In most situations one can freely choose between them. Some differences come from their historical development.

Definition 6.4 A *distance measure* on a set U is a real-valued function

$$d : U \times U \rightarrow [0, 1].$$

A difference to similarity measures is that the range is unbounded. Often, one can restrict to a bounded range. The following properties can hold; they are possible axioms:

- $\forall x: d(x, x) = 0$ (Reflexivity)
- $\forall x, y: d(x, y) = 0 \Leftrightarrow x = y$ (Strong reflexivity)

- $\forall x, y: d(x, y) = d(y, x)$ (Symmetry)
- $\forall x, y, z: d(x, y) + d(y, z) \geq d(x, z)$ (Triangle Inequality)

A distance function is called a pseudo-metric if it is reflexive and symmetric and the triangle inequality holds. It is a metric if in addition the strong reflexivity holds.

Often in mathematics, the range of a distance measure is not bounded; the metrics there are functions:

$$d : U \times U \rightarrow [0, \infty).$$

Such distance functions are mathematically equivalent to certain similarity measures,

$$\text{sim} : U \times U \rightarrow (0, 1] \quad (\text{or } (-\infty, 1]),$$

because “no upper bound” for distances corresponds to “no lower bound” for similarities.

In many applications the intuitively given range is of the form $[0, b]$, where $b > 0$ is a real number. Of course, there is always a simple normalizing transformation of $[0, b]$ to $[0, 1]$. However, with normalizing there is an associated problem. It consists in the difficulty that outliers sometimes are harder to detect. Suppose that for some object y and all objects x one has $d(x, y) \leq 10$ except for one object x_0 where $d(x_0, y) = 50$ holds. This outlier needs special attention and may be erroneous. However, by normalizing the distance measure, one may not recognise the outlier.

Each distance function induces a similarity relation by “ x is more similar (has less distance) to y than to z ”:

$$\text{sim}(x, y) \geq \text{sim}(x, z) \Leftrightarrow d(x, y) \leq d(x, z).$$

From an intuitive point of view, similarity measures and distance functions are equivalent: “very close” is the negation of “far away”. There are, however, two kinds of difficulties:

- What is easily definable for similarity functions may be difficult to express in terms of distances, and vice versa. For instance, the triangle axiom cannot be easily expressed in terms of similarity measures.
- Different kinds of techniques have been developed for similarities and distances. In applications one might be more interested in investigating closeness rather than apartness, or vice versa. As said, this results in methods that faster excludes large distance, or large similarities.

Both similarity measures sim and distance functions d induce similarity relations S_{sim} and S_d in an obvious way.

- $S_d(x, y, u, v) \Leftrightarrow d(u, v) \geq d(x, y)$
- $S_{\text{sim}}(x, y, u, v) \Leftrightarrow \text{sim}(x, y) \geq \text{sim}(u, v)$

There is a natural qualitative relation between measures and distances.

Definition 6.5 A similarity measure sim and a distance measure d are called compatible if and only if:

$$\text{For all } x, y, u, v \in M : S_d(x, y, u, v) \Leftrightarrow S_{\text{sim}}(x, y, u, v)$$

(i.e., the same similarity relation is induced).

This is a qualitative comparison and neglects numerical values. As a consequence we see that two objects that are considered as close together by a similarity measure may be regarded as relatively apart from the point of view of a compatible distance measure. These differences will disappear when more objects are considered.

For comparing similarity measures sim and distance functions d on a quantitative level, one uses transformations of measures that are bijective, order-inverting mappings.

$f : [0, 1] \rightarrow [0, 1]$ with

- $f(0) = 1$
- $f(d(x, y)) = \text{sim}(x, y)$.

Under these conditions sim and d are compatible. Examples are:

- (a) $f(x) = 1 - \frac{x}{\max}$ if a maximal distance \max exists
- (b) $f(x) = 1 - \frac{x}{1+x}$

For converting similarities to distances one can, for instance, take

$$f(x) = \frac{1-x}{x},$$

which gives a distance measure with unlimited range; it is undefined for $x = 0$.

However, such mappings f may have the consequence that the numerical difference based on the similarity is not the same as when they are based on the distances, even if both are compatible.

Example We take the real plane and $d(x, y) = |x - y|$ and $f(x) = \frac{1}{1+x}$. Graphically this is shown in Fig. 6.5.

Here one sees that the numerical differences obtained by d and sim are not the same. From the induced similarity relation we take over the k -nearest neighbour relation.

If only a few objects are given, it is easier for a human to provide a relational similarity than a quantitative one. Humans may not be able to define the similarity relation formally but often they are able for given x, y , and z to decide about the relation $y \geq_x z$. For a large number of objects, a similarity measure is more useful. However, one has to be careful: If one looks at the induced relation of a measure, then it may very well happen that it disagrees with one's intuition, and the measure has to be improved.

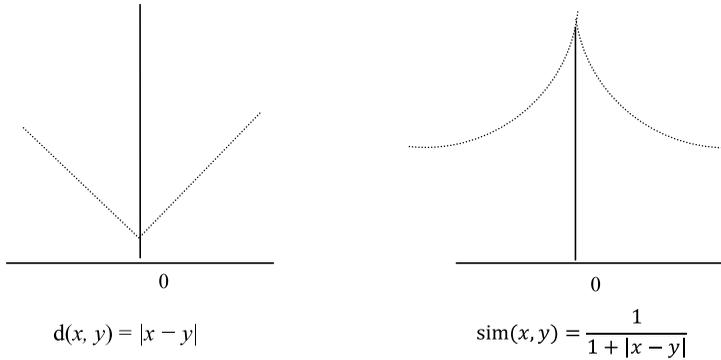


Fig. 6.5 Similarities versus distances

6.4 Types of Similarity Measures

Although there is no universal similarity measure we are still able to distinguish types. For such types, there is no sharp boundary between them. We consider the following types:

- (1) Counting similarities: Certain occurrences in the representation are counted (with possible weights). One can, for instance, count the number of members in a family for tax reasons. This is a weak form of the transformation similarities.
- (2) Metric similarities: They arise as variations of Euclidean metrics. This is closest to the travel view. It is justified if the metric mimics the difference between the object.
- (3) Transformation similarities: Here one measures how costly it is to transform the first object into the second one, for instances, making use of previous plans or correcting misprints.
- (4) Structure-oriented similarities: The structure in which the knowledge is presented plays a role, for instance, in an object-oriented representation. This is also distance-oriented but not in a metric sense; it is rather mostly symbolic. Sometimes it is called path-oriented similarity because it relies on paths of a structure (e.g., taxonomy).
- (5) Information-oriented similarities: The information and knowledge contained in an object plays an essential role. This is natural if one compares texts or images; they can be considered if they provide similar information to the user.
- (6) Relevance-oriented similarities: They weigh importance of different aspects contributing to similarities. This is somehow not a type in its own right; it is rather an add-on to the other types mentioned.
- (7) Dynamic-oriented similarities: They consider and compare dynamic processes.

This shows that there is a variety of measures, and not a unique one. This problem will guide us through the next investigations. For each kind of similarity, precise definitions will be provided.

In some situations the user is interested not only in getting a similarity measure but also in finding out why certain objects are similar, for instance, in a medical diagnosis situation where one has to decide on a therapy based on a diagnosis. In the CBR context this means getting information on why the actual situation is sufficiently similar than a given one.

In other situations the user may emphasize more the distance view: Why is this product not good for me? Why is this therapy not suitable?

This plays a role in deciding the order in which the attribute value attribute-value in the query are exhibited. Certain attributes may be more suitable for rejection and others more suitable for acceptance.

Example Consider the following objects of the car domain:

Car 1: (price = 50,000, #seats = 2, max speed = 150 mph, colour = blue).

Car 2: (price = 20,000, #seats = 4, max speed = 120 mph, colour = red).

Car 3: (price = 20,000, #seats = 2, max speed = 120 mph, colour = red).

Demanded car: (price = 30,000, #seats = 2, max speed = 150 mph, colour = blue).

Importance (e.g., price: high; #seats: low; max speed: medium; colour: very low).

One would ask price first because this may be a reason for rejection; on the other hand it would not be a reason for acceptance on its own. Colour would not be a reason for rejection. Decisions based on these would be:

Car 1 is not good because of price: rejected.

Car 2 is good because of price and #seats: accepted.

All other attributes are not discussed, so the attention of the user is focused on the important aspects. As a result, the user can get a better insight into the decision or may change the demands. Hence, CBR not only provides a solution, but can also give some explanation for it.

In order to discuss these aspects we need a closer look at how similarity measures are defined. We will discuss this in more detail in Chap. 7, Complex Similarity Topics.

It is now time to illustrate the concepts by examples. We start with some simple and popular ones. Here we refer to flat attribute-value representations. Due to their simple structure, the structure of the measures will be simple too.

6.4.1 Counting Similarities

This is the first type we consider.

6.4.1.1 Hamming Measures

A prominent and basic similarity measure is the Hamming measure, coming from coding theory. It applies to attribute-value representations with Boolean-valued at-

tributes:

$$H((x_1, \dots, x_n), (y_1, \dots, y_n)) := \sum (i | x_i = y_i, 1 \leq i \leq n).$$

This measure has the following properties:

$H(x, y) \in [0, n]$ and n is the maximum similarity, which means that $H(x, y)$ is the number of agreeing attribute-value.

H is a symmetric and reflexive:

$$H(x, x) = 0,$$

$$H(x, y) = H(y, x),$$

$$H((x_1, \dots, x_n), (y_1, \dots, y_n)) = H((1 - x_1, \dots, 1 - x_n), (1 - y_1, \dots, 1 - y_n)).$$

The dual is the Hamming distance, which counts the number of disagreeing values.

The scalar (or dot) product is a variation of the Hamming measure but it is essentially the same. It is defined as

$$S(x, y) := \sum_{i=1}^n x_i \cdot y_i.$$

The scalar product is often used in pattern recognition as a simple measure.

For binary attributes we distinguish two cases:

- (a) The values are 0 and 1: Then only values 1 contribute to the accumulated similarity.
- (b) The values are -1 and $+1$: Then, in addition, non-agreeing values give a penalty to the measure.

Both the Hamming measure and scalar product are of very simple character: The only knowledge they contain is the number of coinciding attribute-value.

Here we make a first remark about the possible knowledge contained in a measure. It means that the nearest neighbour obtained by using the measure is really of interest to the user.

This leads to the type relevance-oriented measures refining these measures.

The Hamming measure has little knowledge because it cannot capture the fact that not all attributes are of equal importance based to on their meaning. To overcome this difficulty the weighted Hamming measure was introduced:

$$H_\omega((a_1, \dots, a_n), (b_1, \dots, b_n)) = \sum (\omega_i | a_i = b_i, 1 \leq i \leq n)$$

where

$$\omega = (\omega_1, \dots, \omega_n) \quad \text{with} \quad \sum \omega_i = 1$$

Table 6.2 A counting measure

	Beach	Disco	Tennis	English speaking	Car rental
Query	+	+	+	-	+
Offer	+	+	-	-	-

is a weight vector of nonnegative coefficients. The attribute-value comparisons are still simple (only test for equality) while the weights are intended to reflect the overall importance of attributes. They have the effect that the influence of the attributes for computing the measure is considered.

As already mentioned, it is a major issue to determine the weights and it is one of the first questions everybody asks when a CBR system has to be developed.

6.4.1.2 Tversky Measures

A measure frequently used in cognitive psychology is the Tversky feature model or contrast model. It again generalizes the Hamming measure. A feature here is a Boolean-valued attribute.

We consider the sets P (problems) and S (solutions). Problems as well as solutions are described by the attributes with value 1; in this case we say the feature is observed.

Definition 6.6

- i. $\text{Com}(p, s)$: The set of common features observed for both, $p \in P$ and $s \in S$; $a = |\text{Com}|$.
- ii. $P - C(p, s)$: The set of redundant features observed for p but not for s ; $b = |P - C|$.
- iii. $C - P(p, s)$: The set of missing features observed for s but not for p ; $c = |C - P|$.
- iv. $\text{NO}(p, s)$: Features not observed for p and for s ; $d = |\text{NO}|$.
- v. $S(p, s)$: The set of features observed in s ; $e = |S|$.

The basic idea is that the similarity increases if a feature is common to both objects and decreases if it is observed in one object only. In this context, the different types of objects (problem objects p and case objects c) play a role. This leads to a counting similarity measure.

An example from the vacation domain is in Table 6.2.

From this we can compute the values for the coefficients. Superficially, the example could indicate that the measure cannot be very useful because one has binary attributes only and importance is not considered. This argument is often valid in examples like this. The situation is, however, different in applications where one has hundreds or thousands of features, as in molecule or genome analysis. There is no additional knowledge available and the fact that Tversky measures are efficient to compute plays a role.

The contrast model refines this simple view. The features are split into three subsets (depending on p and s): The contrast model assigns a numerical value $\text{sim}(p, s)$ for the similarity on the basis of these observations. It is defined by a real-valued function h on sets of features and three positive real numbers α , β , and γ :

$$\text{sim}_{\text{contr}}(p, s) = \gamma h(\text{Com}(p, s)) - \alpha h(P - C(p, s)) - \beta h(C - P(p, s)).$$

If the function h is a weighted sum as in the weighted Hamming measure, we get

$$\begin{aligned} \text{sim}_{\text{contr}}(p, s) = & \gamma \sum (\omega_i | i \in \text{Com}(p, s)) \\ & - \alpha \sum (\omega_i | i \in (P - C(p, s))) - \beta \sum (\omega_i | i \in (C - P(p, s))). \end{aligned}$$

The set NO is responsible for the reliability of the measure. If it is relatively large then the measure will not tell much.

For $\alpha \neq \beta$ the measure is not symmetric. In experiments with humans looking at shapes, it is reported that humans prefer $\alpha > \beta$, i.e., the similarity increases with the number of features in the actual problem that are also in the solution, while features observed in the problem object only play a lesser role. It is also reported that for images, diverging features are rated higher while in verbal descriptions of the images, features in Com are rated higher.

Another slight computational variation of the Tversky measure is the Jaccard coefficient. There the comparison of the features uses no longer binary (indicating presence or absence of features) but real values; these measures are called Fuzzy Features Contrast Models.

There are many other variations of such measures. They have many applications in databases, pattern recognition, image interpretation, and biology. There is much experience on when to use which measure, but there is no general theory.

6.4.2 Metric Similarities

They are applicable to attributes with numerical values and closely related to numerical distances. If symbolic values are present they have first to be numerically coded.

In mathematics, metrics are discussed in great detail and many distance functions have been defined for various purposes. We will start with some more elementary examples where we freely use similarities and distances alternatively.

A simple example is the city block metric d_c :

$$d_c(x, y) = \sum |x_i - y_i|.$$

This metric got its name from walking or driving on streets of a city. It sounds realistic but one should realise that it abstracts quite a bit from reality: There may be one-way streets and hilly streets, which are important for the speed of cars and

pedestrians. In the same way, weighted Euclidean measures can be defined more realistically. They are given in the form of distances:

$$d(x, y) = \sqrt{\sum_{i=1}^n \omega_i \cdot (x_i - y_i)^2}.$$

More general is the Minkowski distance:

$$d_{mink}(x, y) = \sqrt[p]{\sum_{i=1}^n |x_i - y_i|^p}.$$

For the real space, a useful type of property often used is the invariance property. A typical example is scale invariance. It means that for a linear transformation T we have

$$d(x, y) = d(Tx, Ty).$$

Scale invariance is not always guaranteed. Suppose we consider the distance of two cities from the perspective of our hometown. The more distant the cities are from home, the smaller their distance may look. In a similar way, rotation invariance is important for image understanding. This means for instance that one can identify the face of a person even if the image is rotated.

6.4.3 Comparisons

Next, we give in Table 6.3 a list of some frequently used elementary similarity measures.

There is a huge number of simple similarity measures that are used in attribute-based CBR. Often, they differ only in the choice of certain coefficients. This choice is motivated by the needs of some specific application. Mostly, the context gives the required insight into use. However, the choice is mostly ad hoc and there is no principle behind it. They are mainly of the type counting similarities.

Next, we give a short overview of measures for attribute-value representations that contain some additional measures besides the ones introduced in Table 6.3.

There are many small variations of these measures. They are useful in one or the other special context but do not provide particularly new general insights. Often, one and the same measure is offered under different names. Some other similarities and their extensions will be discussed next and in Chap. 17, Textual CBR.

6.4.4 Structured Similarities and Symbolic Arguments

Here the case representation is given by attributes with symbolic values. Because the symbolic arguments are often structures, the measure should reflect the structure.

Table 6.3 Elementary measures Euclidean distance

	Formula	Comments
Euclidean distance	$\text{SQRT}(b + c)$	The square root of the sum of discordant objects, minimum value is 0, and it has no upper limit.
Squared Euclidean distance	$b + c$	The sum of discordant cases, minimum value is 0, and it has no upper limit.
Positive evidence	$a/a + b + c$	Only positive features contribute to the similarity.
Pattern difference	$bc/(n^2)$	Dissimilarity measure for binary data that ranges from 0 to 1.
Variance	$(b + c)/4n$	Ranges from 0 to 1.
Simple matching	$(a + d)/n$	This is the ratio of matches to the total number of values. Equal weight is given to matches and non matches.
Dice	$2a/(2a + b + c)$	This is an index in which joint absences are excluded from consideration, and matches are weighted double. Also known as the Czekanowski or Sorensen measure.
Lance and Williams	$(b + c)/(2a + b + c)$	Range of 0 to 1. (Also known as the Bray-Curtis non-metric coefficient.)
Nei and Lei's genetic distance	$2a/[(a + b) + (a + c)]$	Used for describing genetic distances.
Yule coefficient	$(ad - bc)/(ad + bc)$	A function of the cross-ratio; has range $[-1, 1]$. Also known as the coefficient of colligation.
Jaccard, Tanimot	$a/(b + e)$	A kind of default coefficient.

This will play an essential role when we in investigate complex objects such as technical devices.

Symbolic arguments often are coded by numerical values. Such coding is sometimes but not always justified. If one encodes symbolic values into numerical values this increases efficiency at runtime. On the other hand, some transparency is lost because one loses the connection to the original sources of the values. In particular, this creates difficulties in similarity assessment and maintenance, because the original motivations are no longer visible. As a consequence, when one investigates which measure to take it is better to do so at the symbolic level.

Symbolic values can simply be given as a set that can carry a structure. This can be, for example, an ordering or taxonomy. These structures are reflected by measures. Other possibilities are interval-based values. These give a structure when symbolic arguments are coded by numerical values.

For attributes A with unstructured symbolic values $\{v_1, v_2, \dots, v_k\}$, there is no other way for defining measures than using tables; they are also called similarity matrices. Table 6.4 shows a similarity measure, $\text{sim}_A(x, y) = s[x, y]$.

For reflexive similarity measures, diagonal entries are 1 and for symmetric similarity measures we have: upper triangle matrix = lower triangle matrix.

Table 6.4 Symbolic values

$[x, y]$	v_1	v_2	\dots	v_k
v_1	$s[1, 1]$	$s[1, 2]$		$s[1, k]$
v_2	$s[2, 1]$	$s[2, 2]$		$s[2, k]$
\vdots				
v_k	$s[k, 1]$	$s[k, 2]$		$s[k, k]$

The more a structure is defined on the symbolic values, the more systematically can the similarity measures be defined.

For ordered symbols, an ordering is given on the value range.

Example Qualitative values:

{small, medium, large}

where the ordering says

small < medium < large.

We want to use local similarity measures as for numeric types here too. Therefore, we define an integer value for each symbol so that the ordering is preserved. For example:

- small \rightarrow 1
- medium \rightarrow 2
- large \rightarrow 3

On the ordinal scale itself there are no differences between the values defined. The mapping to numbers, however, allows us to model differences between the values. For example, if the gap between small and medium is smaller than the one between medium and large, it is a simple way to represent this knowledge in the measure. Thus, for example:

- small \rightarrow 1
- medium \rightarrow 5
- large \rightarrow 6

would indicate such different gaps, namely, that medium is closer to large than to small.

For classification, the Value Difference Metric (VDM) is frequently used. If one has n classes C_k and the values x, y of an attribute A is given, then we define

$$VDM_A(x, y) = \sum_{k=1}^n |P(A, x, k) - P(A, y, k)|^q.$$

Here $P(A, x, k)$ denotes the conditional probability that the class is C_k provided $x \in \text{dom}(A)$ is given. This probability can be estimated from the case base. The num-

ber q is a constant, usually 1 or 2. The measure can be refined by introducing weighted sums.

A problem arises if an attribute-value representation x occurs in a query that never occurred in any case. Such situations are discussed as “unknown values” in Chap. 13, Advanced Similarity Topics.

Sometimes, one might also combine symbolic with numerical arguments. An example for this is the Heterogeneous Euclidean Overlap Metric (HEOM):

$$HEOM(x, y) = \sqrt{\sum_{i=1}^n d_i(x_i, y_i)^2}, \quad d_i = \begin{cases} H(x_i, y_i), \\ \text{dist}(x_i, y_i), \end{cases}$$

where H is some symbolic measure while dist is the Euclidean metric (one could take any other metric). Again, the measure can be refined by using weighted sums.

6.4.5 Transformational Similarities

In the case of attributes like *name* where the values are strings, the Levenshtein distance is often used, which is of the type transformational measure. It counts the number of changes needed to transform one string into another one. The possible change actions are:

insertion, deletion, and modification.

Essential are only insertion and deletion because modification can be seen as a macro. If these steps have different costs, then more similar means the transformation is cheaper. This measure is of a very simple character but it is sometimes useful for correcting typos. Often it is extended by making use of the fact that certain orderings of letters do not or rarely occur in a language; this is, of course, language dependent.

Transformations occur often if cases are represented as sequences of actions. That happens frequently if the cases are graphs or plans. For example, if one compares technical devices a measure is given by the number (or cost) of changes that transform one device into the other one.

6.5 The Local-Global Principle for Similarity Measures

So far a number of simple measures have been introduced. They are based on structurally simple object representations. For dealing with more complex structures a systematic principle is needed.

In Chap. 5, Case Representations, the local-global principle for general objects and case representations in attribute-based CBR were introduced. It says that each object A is constructed from atomic parts A_i that are attributes by some construction process:

$$A = C(A_i | i \in I).$$

This principle is adequate to a wide range of applications. It is also formulated for similarities and is applied to objects constructed from atomic parts that are described by attributes. It recommends to compare first the atomic parts and then to compare more complex constructs. The comparison of the atomic parts reflects a local view and will be done by so-called local measures. The comparison of the whole objects reflects a global view.

A basic assumption for applying the principle for similarity measures is that both arguments of the measure follow the same construction process. This allows us comparing the corresponding sub-objects in a systematic way.

Formally, this is done in the following way, which is the exact analogue of the construction of complex object representations.

6.5.1 Weighted Measures

Suppose there are local measures sim_i on the domains of attributes A_i and there is some constructor function F such that for $a, b \in U$ (U being the universe under consideration), $a = (a_i | i \in I)$, $b = (b_i | i \in I)$, one has:

$$\text{sim}(a, b) = F(\text{sim}_i(a_i, b_i) | i \in I).$$

One calls the measure sim the global measure. The local measures sim_i compare values of individual attributes and sim compares objects from a global point of view. A particular and frequently occurring as well as a desirable situation is F being a linear sum.

This means sim is of the form

$$\text{sim}(a, b) = \sum_{i=1}^{i=n} \omega_i \text{sim}_i(a_i, b_i).$$

Here the weights ω_i are crucial. They reflect the influence of the attributes on the global measure. For that reason, much attention has been paid to the weights.

For linear weighted measures the similarity increases with the number attributes considered. One has to be careful when dealing with the similarity 0. What does it mean when the local similarity measure for some attribute value is 0? It means only that this attribute contributes nothing to the global similarity. It does not mean that the whole case is excluded from further consideration. However, sometimes this is desired. One can use negative values for representing penalties. But such a filtering will not be done by the similarity measures. It is achieved by retrieval methods and will be discussed in Chap. 8, Retrieval.

When one defines such a measure for complex objects the first step is to look at the local-global representation for the objects. The definition of the measure should observe two conditions:

- (1) The attributes for the measure should be the same as those for the object description.

- (2) The constructor function F should be defined in parallel to the constructor function of the object description.

However, the constructor function needs to contain some additional information. The object description deals with single objects only while the similarity measure compares two objects. As a consequence, the measure has to contain “comparison knowledge”.

A problem for the local-global principle arises because there may be dependencies among the attributes that influence the similarity. In this case the function F has to incorporate the dependencies, which is often quite difficult. In fact, several types of measures have to be excluded in such a situation. For dealing with such problems virtual attributes have to be added; see Sect. 6.6.

6.5.2 Local Measures

The local measures should represent domain properties and their relation to the task in question. This means they are in general not uniform even if the domain is a numerical interval. As an example we consider water, whose temperature in Celsius is an attribute with the domain T , e.g., $T(\text{water}) = [-100, +200]$. There are certain situations in which a measure dominated by physics puts some attention to the difference: Where water is ice, liquid or steam. However, the similarity will change abruptly across these regions.

Such separating values are called landmarks; they are numbers that separate the real axis into intervals (temperature of water: 0 to 100 degrees). The intervals are intended to represent areas in which the objects in question behave qualitatively similarly. Often, these landmarks are facts from physics and not personal relevancies. We note the following facts:

Each interval is when a qualitative region (where the material behaves in some sense very similarly).

The measure is not uniform: numbers inside a region have a high degree of similarity but numbers in different regions have low degrees of similarity. When driving a car on a road, a difference of 4 °C is crucial around 0 °C, but not around 20 °C. One may be interested in other properties of water, which will lead to other local measures.

Now we present some examples of local measures with numerical arguments. For attributes A with integer or real values, similarity is often based on the difference of attribute-value and not on the attribute-value themselves, for example:

- linearly scaled value ranges: $\text{sim}_A(x, y) = f(x - y) = f(d)$
- exponentially scaled value ranges: $\text{sim}_A(x, y) = f(\log(x) - \log(y))$

where

$$f : \mathbb{R} \rightarrow [0 \dots 1] \quad \text{or} \quad \mathbb{Z} \rightarrow [0 \dots 1] \quad \text{and} \quad d = x - y.$$

Here $f(0) = 1$ indicates reflexivity.

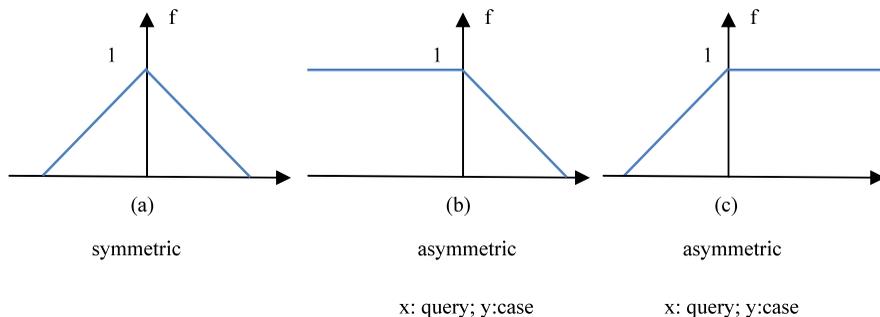
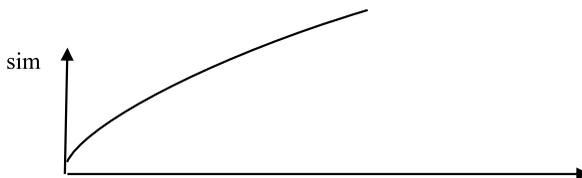


Fig. 6.6 Symmetry and asymmetry

Fig. 6.7 Diminishing utility



Often, one assumes that $f(x)$ is monotonously decreasing for $x > 0$ or monotonously increasing for $x < 0$.

In the following diagrams the x -axis denotes the difference $d = x - y$ and the y -axis denotes $\text{sim}(x - y)$. The use of the difference makes the similarity easier to visualise because we have only one dimension to consider. Some symmetric and asymmetric measures are shown in Fig. 6.6.

The interpretation of Fig. 6.6(b) is “more is perfect” (“the higher the quality, the better”) while Fig. 6.6(c) means “less is perfect” (“cheaper is better”).

The formula for diagram Fig. 6.6(b) is:

$$\text{sim}(q, c) = \begin{cases} 1 & \text{if } d(q, c) < \min \\ \frac{\max d(q,c) - d(q,c)}{\max - \min} & \text{if } \min \leq d(q, c) \leq \max \\ 0 & \text{if } d(q, c) > \max \end{cases} \quad \text{with } d(q, c) = |q - c|; \quad 0 \leq \min < \max \leq \text{Max}.$$

Local measures can be symmetric or asymmetric. Asymmetric means it is important that the attribute-value is in the query or the answer. For price, one is satisfied if the offered price is lower than the required one; the opposite occurs for the quality of a product.

Step functions frequently occur if there is only the possibility that an attribute-value of a case c is useful or not useful with respect to the query-attribute-value q :

$$\text{sim}(q, c) = \begin{cases} 1 & \text{if } d(q, c) < S \\ 0 & \text{else} \end{cases} \quad \text{with } d(q, c) = |q - c|.$$

Instead of linear functions, more complex ones for the decrease or increase can occur that contain more knowledge about the problem, as illustrated in Fig. 6.7.

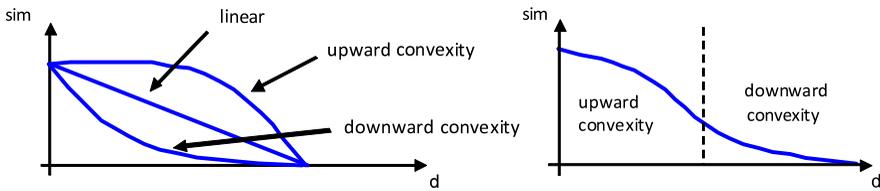
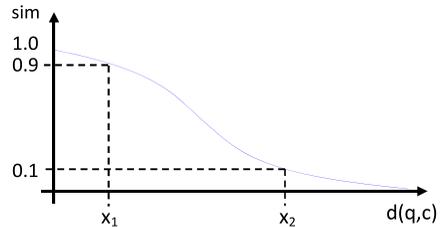


Fig. 6.8 Convexity

Fig. 6.9 Sigmoid function



This curve illustrates the principle of “diminishing utility”: The more one has, the less important is the improvement. In Fig. 6.8 several forms of convexity are shown.

- Downward convexity: A small difference between two values causes a large decrease in similarity.
- Upward convexity: A small difference between two values causes only a small decrease in similarity.

Such knowledge is often decisive for technical machines as well as for economic decisions.

If we tolerate differences between the two values even up to a defined distance x_1 , we can use the following similarity function as shown in Fig. 6.9 (sigmoid function), which gives rise to an even more sophisticated local measure that often occurs in applications:

$$\text{sim} = \exp((-ax)^3) \quad \text{with } a \in \mathbb{R}.$$

As a result, one can say that a local attribute can contain important knowledge for solving the problem. Because the knowledge can be of arbitrary nature, there is no general method to represent it in the measure. This decision depends on the familiarity with the domain and the representation methods.

As an example, we consider the attribute price (for an object, a therapy, and so on). In the problem, x , the desired price, is presented and perfectly matched in the case for $y = x$. In the first situation (a), the utility is still optimal if one has to pay less and in the second situation (b) it is still optimal if one gets more. This justifies the shape of the curves. However, there is a difficulty because the utility function may not be bounded. This means, charging higher prices will involve increasing the utility, and this is not reflected by the measure. From the viewpoint of fuzzy sets, these measures would be regarded as fuzzy preferences.

The different shapes of the curves show that dependency of the measure on the values may be more or less subtle. They reflect the user preferences.

6.5.3 Global Aspects

The constructor function F and the construction operator C for both similarities and distance are closely related. In fact, the definition of the measure should follow the structure of the objects. We will discuss this principle below for various examples. Some very simple examples of the amalgamation C function for measures of attributes with numerical values are:

$$\text{Weighted average: } F(x_1, \dots, x_n) = \sum_{i=1}^n \omega_i \cdot x_i$$

$$\text{with } \sum_{i=1}^n \omega_i = 1 \quad \text{and} \quad x_i = \text{sim}_i(a_i, b_i),$$

$$\text{More general: } F(x_1, \dots, x_n) = \sqrt[\alpha]{\sum_{i=1}^n \omega_i \cdot (x_i)^\alpha} \quad \text{with } \alpha > 0, \sum_{i=1}^n \omega_i = 1,$$

$$\text{Maximum: } F(x_1, \dots, x_n) = \max\{x_1, \dots, x_n\},$$

$$\text{Minimum: } F(x_1, \dots, x_n) = \min\{x_1, \dots, x_n\}.$$

Another simple example is a further generalization of the weighted Hamming measure. It is obtained by allowing arbitrary ranges of the attributes and arbitrary local similarity measures. If $\text{sim} = (\text{sim}_1, \dots, \text{sim}_n)$ is such a measure vector, the generalized Hamming measure is defined as:

$$H_{\omega, \text{sim}}((a_1, \dots, a_n), (b_1, \dots, b_n)) = \sum (\omega_i \cdot \text{sim}_i(a_i, b_i) | 1 \leq i \leq n).$$

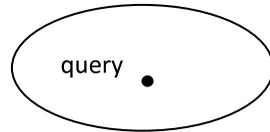
Such measures have been used extensively in the previous examples. It is important to remark that these measures assume the independence of the attributes.

$H_{\omega, \text{sim}}$ is one of the most often used measures. It is very efficient to compute and easy to overlook and understand. A problem is that the summands have to be independent, as mentioned.

6.5.4 Weights

In Chap. 5, Case Representations, we introduced the influence relation and the influence functions. These functions can now be integrated into the similarity measures. They are part of the global measure description. This is simple for attribute-value

Fig. 6.10 Elliptic queries



representations and introduced in Sect. 2.7. The weights should denote the relevance of the different attributes, where relevance indicates the influence for the global measure, i.e., for the indication of what the nearest neighbours are. For the whole case description one should only keep attributes with positive relevance degrees, i.e., with positive weights. The relevance depends on the context, i.e., the purpose for which cases are used. This means that the relevance depends on the goal and the utility.

Consider the car example introduced in Chap. 2, Basic CBR Elements. The attribute “number of doors” was:

- important for selling a car;
- of no relevance for repairing the car.

The weights can also depend on the specific objects under consideration. For instance, for a Ferrari car, the price is less important than for a small Volkswagen. More generally this means that the weights depend on a specific case, to which the measures are applied. This means one is dealing with

$$\sum (\omega_i(B) \cdot \text{sim}_i(a_i, b_i) | 1 \leq i \leq n).$$

Sometimes it is useful to weight attributes higher when one has an exact match rather than an inexact one. For this purpose one can introduce a threshold θ and define

$$\omega_i = \begin{cases} \omega_1, & \text{sim}_i(a_i, b_i) < \theta, \\ \omega_2, & \text{sim}_i(a_i, b_i) \geq \theta. \end{cases}$$

Another popular way of describing weighted measures in the case of real values is by introducing elliptic queries. Traditionally, the nearest neighbours are computed by using circles around the point of interest (the query). If the circles are no longer adequate, the approach uses hyper-elliptic surfaces in order to represent the query. The neighbours are not described by saying how far the objects are from the query, but some dimensions count more than others, as in Fig. 6.10.

However, this is nothing more than introducing weights; it is in one-to-one correspondence to choose weights for the axes of the ellipses.

A concept related to importance is that of independency. This notion exists in various disciplines, ranging from statistics to politics. Here it is appropriate to relate this notion to importance.

Definition 6.7 Two attributes A and B are independent if their importance is not coupled, i.e., the change in the influence of one attribute on the solution has no effect on the influence of the other attribute to the solution.

This implies that one can choose the weights for attributes independently too. It allows studying attributes as single entities. In several situations independency is not present. This is the case if changing an attribute-value changes the value of the result in a certain direction only if the dependent attribute changes its value too. For this reason we cannot assign weights to individual attributes in order to reflect their influence and relevance. Therefore, weighted measures have to be excluded, which is unfortunate. This situation will be discussed below in the context of virtual attributes.

Choosing weights for planning tasks can be quite complex. Consider the example of release planning. Release planning is an early step in planning software projects. The task is to divide the set of features to be done into disjoint packages of limited size that are ordered by discrete time steps. Features in an early release are done before those in a later release. Besides technical restrictions there are customer demands for features and the customers may be of different importance too. This situation occurs often in a more general way. There are k (fixed) boxes of limited size and n objects given and there are m agents who have demands for the distribution of objects among the boxes. The boxes are linearly ordered; this order determines a relation called “better” or “earlier”. The plans generated assign to each object a box to which it belongs.

In order to manage the planning properly, one is interested in how much two results of the planning differ when the demands are changed. This gives rise to a similarity measure between two releases. Suppose there are N features. The global similarity is:

$$\text{sim}(\text{Release plan}_1, \text{Release plan}_2) = \sum_{m=1}^N \frac{\text{sim}_m(\text{Release plan}_1, \text{Release plan}_2)}{N}.$$

The local similarity is:

$$\text{sim}_m(\text{Release plan}_1, \text{Release plan}_2) = \text{abs}(\omega_{m1}\text{Release}(f_m)_1 - \omega_{m2}\text{Release}(f_m)_2).$$

The terms occurring in the formula are:

- $\text{Release}(f_m)_i$ denotes the release of feature f_m in plan i , $i = 1, 2$.
- The weights ω_{mi} denote the costs (the resource consumption) for the company for feature f_m in release i . Here we note that the cost depends on the plan because the consumption can depend on other features in the same release.
- N is the total number of requirements.

It is important to observe that the similarity does not only count the differences. The measure takes also the costs of the requirements into account; they are measured as resource consumptions. In this situation it is useful to put the weights into the definition of the local measures.

Table 6.5 A virtual attribute

Query	Case 1	Case 2	Case 3
Income 2000	Income 1000	Income 2000	Income 6000
Spending 1500	Spending 1500	Spending 5000	Spending 4500
Relation I/S 1.333	Relation I/S 0.67	Relation I/S 0.4	Relation I/S 1.333
Reliable?	No	No	Yes

6.6 Virtual Attributes

We return to the problem of handling dependent attributes. Although such attributes may be sufficient for describing some situation exactly, their importance lies in their combination. Table 6.5 shows an example: Reliability for getting a loan from a bank; similarity depends on the quotient of income and spending.

This means, assigning weights to the individual attributes makes little sense. As a consequence, often a weighted sum measure cannot describe the similarity in an adequate way. In Chap. 13 Advanced Similarity Topics we study the classical XOR problem that shows that it is impossible to find directly a weighted sum measure for solving it. A way out is to introduce additional attributes that reflect the dependencies explicitly. This means they describe the utilities more directly. They are defined in terms of the given attributes and are called virtual attributes.

Example Suppose now the bank wants to compare two persons who want to get a loan. Will they become more similar if we change the income of the first person in such a way that it becomes equal to the income of the second person? In general this is not the case because for each the relation of income and spending is essential.

Suppose by observing the behaviour of the customers the bank has found out:

$$\text{decision: Reliability} = \text{Yes} \Leftrightarrow I/S \geq 1.$$

This is a form to quantify the dependency and gives rise to a new attribute. Note that the decision needs only the virtual attribute I/S !

A large problem is to find and define virtual attributes. They are not sitting in the classroom, raising the finger and saying “Hello, I am missing!” Often one knows there is a certain dependency without being able to formulate an attribute for it. This splits the problem into two subproblems:

- (1) To find out that there is a dependency.
- (2) To quantify the dependency.

A possible approach to the first problem is to consider examples. For instance, one can change the values of some attribute, as in the bank example, by making the values equal for two cases and asking if the cases are more similar now. If not, the reason must be a dependency. This often leads to the discovery of a dependency suggesting a virtual attribute.

The quantification problem is more complex. One can try to attack it by machine learning methods; see Chap. 10, Evaluation, Revision, and Learning.

When virtual attributes are introduced and the measure is extended correspondingly, one has to verify that all the computations for the values of the old attributes can also be performed by the measure. On the other hand, after introducing virtual attributes some of the initial ones often can be omitted.

Virtual attributes allow simpler similarity measures. The banking example illustrates this. Every reader is asked to define an adequate measure for the banking problem without the virtual attributes!

What is achieved by the virtual attributes from a mathematical point of view is a shifting of nonlinear computations from the measure to the attributes. This is again an example of interrelations between the knowledge containers.

6.7 Which Similarity Measure Should I Use?

This question arises for each designer of a CBR system. While the specific methodology of similarity assessment is given in Chap. 11, Development and Maintenance, some general remarks are made here, where we mainly comment on the type of the measure.

We put ourselves in the position where the case structure, the representation language, and the size of case base are already determined; this happens in many applications. This means that the choice of the attributes has been made and the problem of finding virtual attributes has been solved.

Some major influence factors for the choice of the general type of the measure are:

- a. The representation of the cases. This has the greatest influence on the choice of the type of the adequate similarity measure.
- b. The size of the case base. This influences the retrieval time and in turn the similarity measure.
- c. The number of values in the domain of the attributes.
- d. The efficiency needed for retrieval.
- e. Tool support. A good tool can simplify the user effort.

A decision to make is about whether to use numerical or symbolic measures. Of course, this is determined by the representation language. On the other hand, one can code symbolic representations into numerical ones. There are two aspects to consider:

- (1) Does efficiency of similarity computation play a major role? This aspect is related to coding.
- (2) Is transparency of the representation important? This would suggest keeping the symbolic representation if immediate understanding is desired. This is of particular importance if humans interact.

A basic requirement as earlier mentioned is the compatibility between case representation and the measurement structure. For a flat representation the local-global principle plays no role. The measures taken are mostly metric and counting measures. The major point is efficient retrieval, which is mostly determined by size of the base and the number of attributes.

For complex representations one should use the local-global principle. This means one first analyses the structure of the representation. The next steps are:

- What are the local measures? This requires an understanding of the domain of the attributes and its relation to the task to be solved. The problem is basically to find qualitative regions where the interesting properties of the domain change little.
- What is the global measure? If one likes efficient handling, one likes weighted linear measures. This leads to the following questions:
 - Are the attributes independent or do we need additional virtual attributes?
 - What are the weights? This is difficult to answer. As stated before, they should reflect the importance or relevance of an attribute. However, this delays the answer because we do not know what these are. This question is not only of theoretical interest; everyone who wants to build a CBR system confronts it.

The two major sources of knowledge in this respect are:

- Human expertise: One can ask experts. They usually provide qualitative weights. Often, this is sufficient, for instance, when the input variables are not precise or unreliable.
- Automatic tools for machine learning. Machine Learning methods can also be used for improving qualitative measures given by experts. See Chap. 10, Evaluation, Revision, and Learning.

If the weights reflect importance, one asks: Importance for what? For finding a good solution! This means attribute A is more important than attribute B if the solution found by using just A is better than that found by using just B . Though application dependent, better can mean different things, e.g.,

- The probability for finding the correct solution is higher;
- The expected costs of false solutions are lower;
- and so on.

These aspects should be reflected in the weights. A simple example is given for classification, for which we will give a method for weight computation. To motivate our approach we assume that the attributes are given by persons who answer the question about which class some object is an instance of. Suppose now that Bill outperforms all other persons with his classifications. Would we really take the average of all participating persons? Definitely not! We would give Bill a higher preference and this will result in a higher weight. We will employ this idea in the following.

Definition 6.8 Suppose A is an attribute, a is some value of A , C is the class of the object O under consideration, Exp means expected value.

Table 6.6 Relevance matrix

w_{ij}	A_1	\dots	A_n
case ₁	w_{11}		w_{1n}
\vdots			
case _k	w_{k1}		w_{kn}

- i. $\text{PredictivePower}(A, a, O) = \text{Prob}(C | \text{value}(A) = a)$
- ii. $\text{ExpPredPower}(A, O) = \text{Exp}(\text{PredictivePower}(A, a, O) \times \text{Prob}(a) | a \in \text{dom}(A))$
- iii. $\text{PredPow}(A) = \text{Exp}(\text{PredPower}(A | O) \times \text{Prob}(O))$.

This can be generalized to cost-sensitive classification by counting the costs too. A further generalization is when the error of a solution is quantified.

A way for choosing the weight $w(A)$ for an attribute A is

$$w(A) = \text{PredPow}(A).$$

This means the weight says how often the attribute scores.

In the lucky case the probabilities can be estimated from the given examples. Often, however, the probabilities are fairly difficult to estimate. Then human experts can be helpful. This is comparable with the situation where one has a number of persons who can predict some events. Those who are more often right get more attention and are considered more important.

In many situations this view is not sufficient, but it has the advantage of being simple and easy to implement. We return to this problem in Chap. 10, Evaluation, Revision, and Learning. One way to proceed is to define the values explicitly. Table 6.6 shows for a limited number of cases $\{c_1, \dots, c_k\}$ a relevance matrix.

Often, the assumptions for the computations mentioned are not given and expert knowledge is also not available. Then one can try to learn the weights from examples, if examples are available. This will be discussed in Chap. 10, Evaluation, Revision, and Learning. In any event, tool support is needed.

6.8 Tools

Most general CBR tools allow the use and definition of a variety of measures. A library containing several similarity measures is the SimMetrics Similarity Metric Library (<http://sourceforge.net/projects/simmetrics/>).

A general tool for detecting some dependencies is in Weka (<http://www.cs.waikato.ac.nz/~ml/weka>). Read about it in Hall et al. (2009).

Highly specialised tools exist for random variables, texts and images. They will be discussed in the respective Chaps. 17 and 18.

6.9 Chapter Summary

This chapter gives a general overview of different basic concepts connected with similarity. A major part is devoted to mathematical models of similarity. The difference between relational and functional similarity is discussed. Relational models allow the definition of the important nearest neighbour concept. The relation to distance functions is pointed out, as are several possible axiomatic properties. There is a variety of elementary measures listed that differ only partially but are used frequently.

Some types of measures are introduced. Some elementary measures are counting measures, such as the Hamming similarity and the measures related to metrics. Another type of measures is structure-oriented. It refers mainly to attributes that have symbolic values from which an attribute-based structure is built. The transformational similarities count the steps necessary for transforming objects to objects.

These approaches are unified by introducing the local-global principle from case representations for similarity measures too. This principle relates object representations and measures.

For local measures, first attributes with numerical values are considered. These measures can contain important knowledge about the problem.

In constructing a similarity measure, a basic difficulty is finding the values for weights. The approach presented formalizes weights on the basis of their predictive power. This approach is also used for defining the independence of attributes.

The problem of dependent attributes is approached by introducing virtual attributes.

6.10 Background Information

Similarity has a long history and plays a role in many disciplines that fall outside the scope of this book. An early and very comprehensive reference for simple measures is in Bock (1973). Many of the simple measures have been created for purposes of pattern recognition, for which they are heavily used. An overview can be found in Richter (2007). The nearest neighbour concept plays a role in many areas of mathematics and computer science.

The Levenshtein measure was introduced in Levenshtein (1966). The Tversky contrast model was created in the context of psychology and reported including experiments in Tversky (1977) and Tversky and Gati (1978). Fuzzy Features Contrast Models are discussed in Santini and Jain (1999). They indicate presence or absence of features.

Distance measures are standard concepts in mathematics. Distances can be defined on any domain and they are used for approximation purposes. Because of this they are closely connected to partial orderings. They play a major role in pattern recognition too.

The determination of the weights is part of the quality criterion for conceptual clustering in the system Cobweb; see Fisher (1987). More information on weights is also in Ruhe (2010).

6.11 Exercises

Exercise 1 Find two similarity measures sim_1 and sim_2 for

$$u = (1, 1, 0), \quad v = (0, 1, 1), \quad w = (1, 0, 1)$$

such that

$$\text{sim}_1(u, v) < \text{sim}_1(u, w) \quad \text{and} \quad \text{sim}_2(u, w) < \text{sim}_2(u, v).$$

Exercise 2 Construct a graphical representation for the two-dimensional case with two numerical attributes for visualising the differences between

1. City block metric
2. Euclidean distance
3. Maximum norm

Do this for

- (a) No weights
- (b) Weights $\omega_1 = 0.75$ and $\omega_2 = 0.25$.

Exercise 3 Consider three real-valued attributes A, B, C . One vector (a, b, c) should be better than another vector if $a(b + c) > e(f + g)$.

Is there a weighted similarity measure that reflects this? How can you simplify the problem by introducing a virtual attribute?

Exercise 4 Compute the Hamming distance and the Jaccard coefficient for the vectors

$$x = [0101010001] \quad \text{and} \quad y = [0100011000].$$

Which of the measures is closer to simple matching?

Exercise 5

- (a) Discuss how you might map similarity values from the interval $[-1, 1]$ to the interval $[0, 1]$.
- (b) Given a similarity measure with values in the interval $[0, 1]$, describe two ways to transform this similarity value into a distance value in the interval $[0, \infty]$.

Exercise 6 (for multilingual students) Take any paragraph of this book and translate it into two languages A and B of your choice.

Decide from the view of transformational similarities which language is closer to English, A or B ?

Exercise 7 Write an algorithm for a classification problem that computes the weights for a case base.

A cost-sensitive classification problem is one that assigns costs to misclassifications for each class. The goal is to minimize the overall costs. Extend now the exercise to cost-sensitive classes.

Exercise 8 Suppose you are going on a business trip and two persons can do the packing of your luggage, which has limited space, and you cannot fill everything in that you want. Describe the two packing methods by a detailed measure to find out which person you prefer.

References

- Bock WJ (1973) Philosophical foundations of classical evolutionary classifications. *Syst Zool* 22:375–392
- Fisher DH (1987) Knowledge acquisition via incremental conceptual clustering. *Mach Learn* 2:139–172
- Hall M, Frank E, Holmes G et al (2009) The WEKA data mining software: an update. *SIGKDD Explor Newsl* 11(1):10–18
- Levenshtein VI (1966) Binary codes capable of correcting deletions, insertions and reversals. *Sov Phys Dokl* 10(8):707–710
- Richter MM (2007) Similarity. In: Perner P (ed) *Case-based reasoning for signals and imaging*. Springer, Berlin, pp 25–90
- Ruhe G (2010) *Product release planning: methods, tools and applications*. CRC Press, Boca Raton
- Santini S, Jain R (1999) Similarity measures. *IEEE Trans Pattern Anal Mach Intell* 21:871–883
- Tversky A (1977) Features of similarity. *Psychol Rev* 84:327–352
- Tversky A, Gati I (1978) Studies of similarity. In: Rosch E, Lloyd BB (eds) *Cognition and categorization*. Erlbaum, Hillsdale, pp 79–98

Chapter 7

Complex Similarity Topics

7.1 About This Chapter

Readers of this chapter are expected to have a basic understanding of CBR, which can be obtained in Part I. Reading the previous chapter, Basic Similarity Topics, is recommended. Readers can select certain sections depending on their application interests and on the chosen representation methods. This chapter extends the methods discussed in the previous chapters to mainly structural similarities tailored to more complex object representations.

7.2 Graph Representations and Graph Similarities

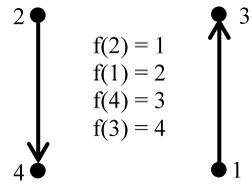
Graphs were introduced in Chap. 5, Case Representations. They are the simplest of the ordered structures we consider that may be annotated or attributed. In all graph representations, structure- or transformation-oriented similarity measures dominate. Some first similarities on graphs compare graphs and distinguish between different graph matching approaches:

- Graph isomorphism.
- Subgraph isomorphism.
- Largest common subgraphs, so-called cliques.

The graph matching approaches initially consider exact matches of the graphs or their subgraphs only. In addition, there is also inexact graph matching, which is more in the spirit of similarity measures. It requires the introduction of graph edit distance, which is used for approximating graphs.

7.2.1 Graph Isomorphism

The graph isomorphism is defined as follows.

Fig. 7.1 Isomorphic graphs**Definition 7.1**

- (i) Two graphs $G_1 = (N_1, E_1)$, $G_2 = (N_2, E_2)$ are isomorphic if there is a bijective mapping $f : N_1 \rightarrow N_2$ such that $(v, w) \in E_1$ if and only if $(f(v), f(w)) \in E_2$.
- (ii) For attributed graphs it is in addition required that the marks α and β on the nodes and the edges coincide, i.e., $\alpha(v) = \alpha(f(v))$ for all $v \in N_1$ and $\beta((v, w)) = \beta((f(v), f(w)))$ for all $(v, w) \in E_1$.

A radical way is to consider two graphs as similar if they are isomorphic. It is only a binary predicate, as discussed in the context of relational similarities, which misses the approximation character of similarity—this is a disadvantage. It says that two objects are similar if they are identical. Another disadvantage is that this testing is in general very complex. Figure 7.1 shows an example of two isomorphic graphs.

7.2.2 Subgraph Isomorphism

Subgraph isomorphism is somewhat more general.

Definition 7.2 A graph G_1 is subgraph isomorphic to a graph G_2 if there is a subgraph G'_2 of G_2 which is isomorphic to G_1 .

Again, one can declare a binary similarity predicate by saying that G_1 is similar to G_2 if G_1 is subgraph isomorphic to G_2 . This is an asymmetric similarity predicate that is difficult to compute.

7.3 Largest Common Subgraphs

A more refined view looks at subgraphs that two graphs have in common. The size $|G|$ of a graph G is the cardinality of the nodes plus the cardinality of the edges.

Definition 7.3 A graph G is a largest common subgraph of G_1 and G_2 if and only if

- i. G is subgraph isomorphic to G_1 and subgraph isomorphic to G_2 ,
- ii. There is no graph G' that is subgraph isomorphic to G_1 and G_2 such that $|G'| > |G|$.

This allows a numerical similarity measure:

$$\text{sim } G_{\text{graph}}(G_1, G_2) = |G'| \quad \text{where } G' \text{ is a largest common subgraph of } G_1 \text{ and } G_2.$$

7.3.1 Edit Operations

In Chap. 6, Basic Similarity Topics, Sect. 6.4.4, on symbolic arguments, we mentioned the Levenshtein measure as an example of a transformational measure. It also plays a role for graphs. The measure looks at possible operations that transform one graph into another one.

These are the edit operations:

- Inserting nodes.
- Deleting nodes.
- Inserting edges.
- Deleting edges.
- Replacements.
- Changing marks on nodes and edges.

The intuition is that graphs are more similar the easier or cheaper the transformations are. This is useful if the operations have costs. It is done by assigning a cost $c(e)$ to some transformation (or edit) operation e .

Edit operations can be ordered in edit sequences (e_1, \dots, e_n) where the costs are added up. Now one can define the edit distance as:

Definition 7.4

$$d(G, G') = \min\{c(s) \mid s = (e_1, \dots, e_n) \text{ and } e_n(\dots(e_1(G))\dots) = G'\},$$

$$c(s) = \sum_{i=1, \dots, n} c(e_i).$$

If the costs of insert and delete operations differ, the edit distance is not symmetric.

Example We consider the two words “combine” and “combination” and the costs $\text{cost}(\text{insert}) = 2$, $\text{cost}(\text{delete}) = 1$, $\text{cost}(\text{replace}) = 1$. Then we get $d(\text{“combine”}, \text{“combination”}) = 9$ because of replacement $e \rightarrow a$, insertion t, i, o, n and $d(\text{“combination”}, \text{“combine”}) = 5$ because of replacement $a \rightarrow e$, deletion t, i, o, n .

Essentially, edit distances are a combination of structural and transformational similarities. If the graph is a linear sequence S , i.e., if there are n linearly ordered objects, a very simple distance is the permutation distance:

Definition 7.5

$d_{perm}(S_1.S_2)$ = minimal number of permutations to transform S_1 into S_2 .

Sometimes one is interested in the overall similarity a set of graphs has. If each graph in the set represents a case, then this shows the overall similarity of the case set. One way to view this is to look at the largest substructure the graphs have in common. This can be called the *coherence* of the graph or case set. In CBR this allows us to partition the case base. One can first split the case base into coherent subbases. In the retrieval phase one can first select such a subbase and then use the search in that subbase for fine tuning. This is an example of a two-phase retrieval; see Chap. 8, Retrieval.

For trees this can be defined as follows. Suppose we have a set of trees:

$$T = \{T_i = (N_i, E_i) | i \in I\}$$

with marked nodes and a common root r for all trees. We consider:

$$Int = \left(\bigcap N_i, \bigcap E_i | i \in I \right).$$

The prototype of T

$$prot(T) = (N_{prot}, E_{prot})$$

is defined as the largest component of Int that contains the root r . A component is a maximal connected subgraph where connected means that all pairs of nodes are connected by some path. The prototype can be seen as a case that represents the whole set in the best way.

The group similarity sim_{group} of T compares all cases in the set with the prototype and is now defined as:

Definition 7.6

$$sim_{group}(T) = \begin{cases} 0 & \text{if } |E_{prot}| = 0, \\ \frac{|E_{prot}|}{|E_T|} & \text{otherwise.} \end{cases}$$

The group similarity is 0 if the prototype has no edges, i.e., it consists of the root only. One can make use of high group similarity of a set of cases in different ways.

One situation occurs when the set of nearest neighbours has a high group similarity. Then one can retrieve the prototype only. Another situation occurs in the context of adaptation.

A dual of group similarity is group dissimilarity that can be defined as

$$dissim_{group}(T) = 1 - sim_{group}(T).$$

It measures how diverse the set of cases is. This concept will be generalized to the concept of diversity in Chap. 14, Advanced Retrieval, where the application of this concept is also discussed.

7.4 Taxonomic Similarities

Taxonomies are widely used structures and specialisations of graphs. They relate objects to each other. The intention is to do this in a hierarchical way, going from general to more specific objects. This implies that branching leads to objects that have more in common. Taxonomies contain formal and informal elements. The graph structures are formal while the objects in the annotations are informal. Because of the information they contain it is no surprise that taxonomies can be used for creating similarity measures. An ideal situation is when the graph structures are isomorphic. This can be weakened in various ways, for instance, as shown for general graphs. In addition, one can make use of the special graph structure in taxonomies. Taxonomies can be used in two ways. One way is to organise objects directly. Another way is to structure the sub-objects or the views of an object. This is the concern here. For taxonomic structures like trees (and in the same way for more general graphs) one can define structural similarities for representing knowledge about the structure. The structure-based similarity relates objects independent of their contents, based on the position in the taxonomy only. However, this is meaningful only if the taxonomy reflects semantic relations, for instance, two successor nodes of a node have something in common.

Inside a taxonomy, the similarity of concepts may be explicitly or implicitly defined by using a taxonomic view of the hierarchy. These similarities make sense only if the taxonomy reflects a meaningful hierarchy for the user. They are useless if the annotations on the nodes are just arbitrary, for instance, “shape” and “volume”, if there is no relation between them.

The simplest distance measure is the graph distance:

Definition 7.7 Given two nodes u and v , the graph distance is

$$d_{gr}(u, v) = \text{length of shortest path from } u \text{ to } v.$$

This is the path through the deepest common predecessor of u and v . This predecessor is called DCP (Deepest Common Predecessor).

This gives rise to the Wu–Palmer metric. It measures the depth of two given concepts in the taxonomy:

Definition 7.8

$$\text{sim}_{WuPal}(c1, c2) = \frac{2 \cdot \text{depth}(\text{DCP}(c1, c2))}{\text{depth}(c1) + \text{depth}(c2)}.$$

Such similarities are path-oriented. We will see extensions of this in Chap. 17, Textual CBR.

In taxonomies one has in general to distinguish the different types of nodes, leaf nodes and inner nodes. For getting leaf nodes we go down starting from the root. If we have a query and the solutions are taxonomic objects, this means in principle

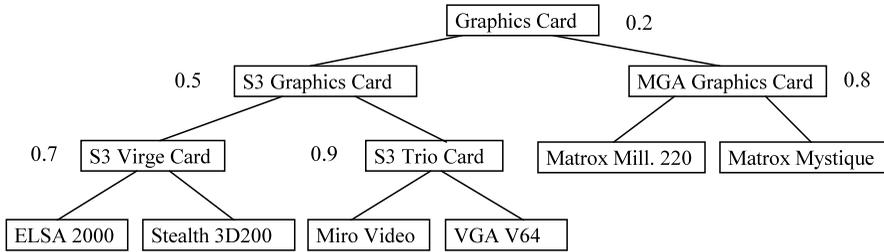


Fig. 7.2 Taxonomy example

that we assign a similarity value to inner nodes with the intention of this indicating the common similarity of all successor nodes. Therefore, if we go down the taxonomy, similarity values are larger because we compare fewer objects. The similarity between two leaf nodes is computed by the similarity value at the *deepest common predecessor*. This means that the similarity of leaf nodes depends only on the position in the taxonomy.

Consider a CBR system for the sales support of personal computers. A case represents an available PC from the stock. The case representation contains an attribute “graphics card” and a taxonomy. As an example, consider the taxonomy describing a small part of a hierarchy of computer tools in Fig. 7.2.

Now we look into the taxonomy at a case C_1 with the *ELSA2000* card and a case C_2 with the *Matrox Mystique* card. If we assume that a customer enters a query to our hypothetical CBR system in which she specifies that she wants a *Miro Video* graphics card, then C_1 is certainly closer to the query, because *Miro Video* and *Elsa2000* have more in common (e.g., the S3 chip) than the *Miro Video* and the *Matrox Mystique*. In general, we could use a similarity measure that assesses similarity based on the distance between case and the query value in the taxonomy tree.

Numerically:

We see: $\text{sim}(\text{ELSA 2000}, \text{Stealth 3D}) = 0.7$ and $\text{sim}(\text{ELSA 2000}, \text{VGA V64}) = 0.5$;

Therefore, *ELSA 2000* is more similar to *Stealth 3D* than to any of the S3 Trio Cards or the MGA Cards.

There are different ways to interpret the similarity for the inner nodes, as we see in the following examples.

- (1) A customer wants a computer with an S3 Graphics card.
Which one does not matter (“do not care semantics”).
- (2) There is a fault for all S3 Graphics cards.
More general case: S3 Graphics card.
- (3) In a diagnostic situation one knows that the user has an S3 Graphics card but it is not known which one (“unknown value”).

For handling of “do not care” we do not have to distinguish the cases where the inner node is in the query q or in the case C .

Next, we use the following notation:

L_q, L_C : Leaf nodes below an inner node q or C , respectively,

$c < Q$: c is in the taxonomy below Q (c is successor of Q),

$S_{(q,c)}$: Similarity value of the least common ancestor q and C .

- (a) The inner node is in the case C and represents a set of values one of which is a correct value for the case. Therefore, the use of this set in the attribute is a shortcut for the use of several cases, one for each value in the set. Since we are looking for the most similar case in the case base, the similarity between the query and our case containing the inner node should be the highest similarity between the query and one of the values from this set. This leads to the next definition.

Definition 7.9

$$\text{SIM}(q, C) = \max\{\text{sim}(q, c) | c \in L_C\} = \begin{cases} 1 & \text{if } q < C, \\ S_{(q,c)} & \text{otherwise.} \end{cases}$$

This definition ensures that the similarity is the same as the similarity that arises when each of the cases with leaf node values would have been stored in the case base. Next, we have to distinguish different possibilities.

- (b) The inner node is query q . Here, the inner node can be viewed as a shortcut for posing several queries to the system, one for each of the values from the set that the node represents. Since we are again interested in the most similar case, we can again select the most similar attribute-value from the set. This leads to the following definition.

Definition 7.10

$$\text{SIM}(q, c) = \max\{\text{sim}(q', c) | q' \in L_Q\} = \begin{cases} 1 & \text{if } c < q, \\ S_{(q,c)} & \text{otherwise.} \end{cases}$$

- (c) The inner node is in both, the query and the case. Since we are looking for the nearest neighbour from the query as well as the case set and since both represent alternatives that are suited equally, this gives rise to the following definition.

Definition 7.11

$$\text{SIM}(q, C) = \max\{\text{sim}(q', c) | q' \in L_Q, c \in L_C\} = \begin{cases} 1 & \text{if } C < q \text{ or } q < C, \\ S_{(q,c)} & \text{otherwise,} \end{cases}$$

$$\text{SIM}(q, c) = \max\{\text{sim}(q', c) | q' \in L_Q\} = \begin{cases} 1 & \text{if } c < q, \\ S_{(q,c)} & \text{otherwise.} \end{cases}$$

The approach of just counting the number of steps to the deepest common predecessor was refined here. The difficulty is that edges are weighted and not all edges

may have the same weights and counting the links will not suffice. Among the generalizations one finds information content measures or additional information about the part-of or the is-a relation.

An interesting point is that one can use context information too. In Chap. 12, Advanced CBR Elements, ontologies are introduced. Now concepts may occur in several task ontologies, e.g., in a chemical as well as in a biological ontology. The context is the specific ontology that can lead to different distances. One can regard this as a utility orientation because an ontology is built for a specific purpose.

Special attention has to be paid to inner nodes with unknown (missing) attribute-value.

“Unknown” semantics: The set contains alternatives that are not all equally acceptable, and this is unknown. Here we have to distinguish whether the missing information is in the query or in the solution. As in the taxonomies, there are different strategies:

(1) Pessimistic strategy:

- a. Missing values in the solution C : $SIM(q, C) = \min\{\text{sim}(q, c) | c \in C\}$
- b. Missing values in the query Q : $SIM(Q, c) = \min\{\text{sim}(q, c) | q \in Q\}$
- c. Missing values in both: $SIM(Q, C) = \min\{\text{sim}(q, c) | c \in C, q \in Q\}$

For the pessimistic strategy we take $\text{sim} = 0$, i.e., we take the lowest possible value.

(2) Optimistic strategy: Take $SIM = 1$, i.e., take the highest possible value.

(3) Probabilistic strategy: Take SIM as the expected value.

We remark here that we make use of these three possibilities in analogous situations several times.

7.5 Similarities for Object-Oriented Representations

The problem of computing the similarity of complex objects has two aspects. One is based on the attributes defining the object and one is based on the classes in which those objects are located.

In an object-oriented representation relational attributes (for instance, “part-of”) and attributes with several values are present. The class structure dominates the object-oriented representation. This structure is as in taxonomies, detailed in the previous section.

In general, the similarity computation between two objects can therefore be divided into two steps:

- the computation of an intra-class similarity SIM_{intra} ,
- and the computation of an inter-class similarity SIM_{inter} .

(1) Intra-class Similarity:

The common properties of the two objects can be used to compute the intra-class similarity. For this we take the most specific common class of the two objects and

compute the similarity based on the attributes of this class only. By our considering the attributes of the most specific common class only, the object similarity computation can be done as usual, since the objects being compared are from the same class. That is, local similarities or object similarities are computed for all attributes and the resulting values are aggregated to compute the intra-class similarity, for instance, by a weighted sum. That means, according to the local-global principle:

Definition 7.12

$$\text{sim}_{intra,c} = C(\text{sim}_1(q, A_1), \dots, \text{sim}_n(q, A_n)),$$

where C is the construction operator, and the sim_i are the local similarities.

(2) Inter-class Similarity:

The intra-class similarity alone would not be sufficient for describing the similarity of two objects because not all relevant properties can be described this way. Therefore, we have to consider other classes too.

The inter-class similarity represents the highest possible similarity of two objects, independent of their attribute-value, but dependent on the positions of their object classes in the hierarchy.

The inter-class similarity

$$\text{sim}_{inter}(Q, C)$$

is defined over the classes of the objects from the query and case to be compared. The deeper one descends in the class hierarchy, the more the number of values the instances of the classes have in common. We can therefore view the inter-class similarity as a measure of how many values the compared objects have in common.

The final object similarity $\text{sim}(q, c)$ between a query object q and a case object c can then be computed by the product of the inter- and intra-class similarity. This is analogous to the similarity in taxonomies. Now we will present a typical example of inter-class similarity.

The first step is always concerned with the local similarities between the attributes. Then we proceed as follows.

Suppose two objects o_1 and o_2 are given and $\text{class}(o_i)$ denotes the class of object o_i . Let

$$\langle \text{class}(o_1), \text{class}(o) \rangle$$

be the most special common super class of the two classes with some measure

$$\text{sim}_{\langle \text{class}(o_1), \text{class}(o_2) \rangle}.$$

This measure will be the measure similarity sim_{inter} .

On the class hierarchy we assume a similarity measure sim^* in the same way as on taxonomies; the semantics of the inner nodes have to be fixed (“unknown” or “do not care”); $\text{sim}^*(\text{class}(o_1), \text{class}(o_2))$ says how similar the two objects can be at most.

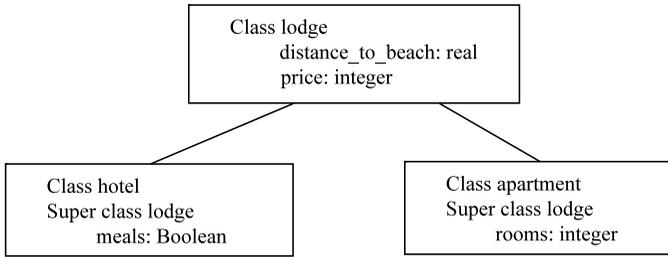


Fig. 7.3 Different classes

Table 7.1 Some similarity values

		Demand	$\text{sim}_{(class(o_1), class(o_2))}(o_1, o_2)$
Hotel	Apartment	0.6	sim_{lodge}
Hotel	Hotel	1	sim_{hotel}
Hotel	Lodge	1	sim_{lodge}
Apartment	Lodge	1	sim_{lodge}
Lodge	Hotel	1	sim_{lodge}

Now we can compute the similarity as

$$\text{sim}(o_1, o_2) = \text{sim}^*(class(o_1), class(o_2)) \cdot \text{sim}_{(class(o_1), class(o_2))}(o_1, o_2).$$

This considers only common inherited attributes.

One sees a specific example in Fig. 7.3, where we consider the following classes for lodging, hotel, and apartment; we have taken the semantics as “arbitrary”.

We see example similarity values for this situation in Table 7.1.

7.6 Many-Valued Attributes

In object-oriented representations one frequently has many-valued attributes. A many-valued attribute contains several attribute values or objects in its domain, which leads to the problem of defining similarities between sets. This is a simple instance of the local-global principle.

Suppose the value sets are

- $A = \{a_1, \dots, a_n\}$ in the case,
- $B = \{b_1, \dots, b_m\}$ in the query.

We suppose the individual similarities $\text{sim}(a_i, b_j)$ are known.

For the set similarity sim we proceed again according the local-global principle:

$$\text{sim}(B, A) = F(\text{sim}(a_1, b_1), \dots, \text{sim}(a_1, b_m), \dots, \text{sim}(a_n, b_1), \dots, \text{sim}(a_n, b_m)).$$

The problem of defining F properly remains as always; it depends on the specific situation. We encounter the same possibilities for the interpretation as in taxonomies:

- (a) “Do not care” semantics: There are alternatives with no differences but one has to make a choice.
- (b) “Unknown” semantics: The set contains alternatives that are not all equally acceptable and this fact is unknown.

Due to the lack of specific knowledge we have different strategies:

- (i) Pessimistic strategy: Choose $F = \min$
- (ii) Optimistic strategy: Choose $F = \max$
- (iii) Probabilistic strategy: Choose the expected value $F = \text{average}$

In a situation with a query q and a case c one can also proceed in the following way:

$$\text{sim}(q, c) = \frac{1}{|\text{dom}(q)|} \cdot \sum_{x \in \text{dom}(q)} \max\{\text{sim}(x, y) | y \in \text{dom}(c)\}.$$

Another way to treat relational attributes is to consider them as “ordinary” attributes and to iterate them recursively for the subclasses.

7.7 Similarity for Processes and Workflows

Processes and workflows can be described as annotated graphs. They carry, however, in general, a very rich structure. The annotations can make use of any of the structures aforementioned and therefore there is no unique similarity measure.

Processes have workflows as instances. For CBR we have demands (problems) and solutions. The demand states what the tasks of the workflows are and the solutions give the workflows. Here we describe both problems and solutions because it is of interest for similarity. These problems are of a rather complex nature. This area is very broad, and we restrict our discussion to software development processes. Even within the scope of software development processes, we can discuss the problems only partially.

The goal of each workflow system is to model and execute processes of the real world. The workflow definitions are executed in concrete situations. The context of the workflow describes situations. Each step in a workflow is a planning step. Basic elements of a workflow are tasks that have to be performed; they are annotations of the nodes of the graph.

The graph ordering in a process usually reflects, among others things, a time ordering. This can have two origins:

- The time ordering records what occurred in temporal order.
- The ordering reflects precedent relations that are mandatory.

Ordering measures will be given in Sect. 7.7.1 on time series. For similarity relations and reuse, this plays a role: accidental temporal relations can be neglected while mandatory relations have to be obeyed in reuse under all circumstances.

The workflow description should contain the elements that are important for reuse. Useful representation structures are:

- Context representation:
 - Task context
 - User context
- Task context:
 - Causal aspects: task goals
 - Operational aspects: applications and tools used
 - Chronological aspects: timeline of events that have occurred in the system, for instance, recently processed workflow tasks
- Task descriptions:
 - Functionality
 - Task implementation methods, and so on.
- Attributes describing the control flow relationships between the tasks.
 - How are the tasks arranged: in sequence, in parallel, or in loops?
 - Mandatory precedence constraints
- A set of parameters required for the execution
- User context:
 - Estimated user goals
 - Organisational structures involved in or relevant to the task: persons, roles, skills, and interests of a user, projects and organisational units
 - Behavioural aspects: the behaviour of the user-performed operations and actions.

These description elements require quite different representation forms ranging from simple attributes over taxonomies to complex objects. From the similarity point of view the heterogeneous data structure makes the subject interesting and difficult. An immediate consequence is that one type of measure will not be sufficient. There are different types of structures and the similarities have to be chosen accordingly.

The problem is how to organise this. A major point is that the types of problem representations and therefore the types of similarity measures are not always compatible with the content-oriented distribution of the workflow description.

A consequence is that besides the workflow representation one needs an internal (and hidden) pool of similarity measures and a method that selects a measure in the pool that determines the local measure to each represented object. Some of the objects, namely, those that are subject to a possible adaptation, should be of transformational type.

Special situations arise when two cases with their derivations, as in derivational analogy (see Chap. 12, Advanced CBR Elements), are compared. A natural similarity measure would tell us how difficult or costly it is to transform one derivation into the other one. This measure would make use of sequences and be asymmetric.

However, this does not reflect the problem situation of the user. The user has a problem and a base of cases.

A possible approach would be:

- (1) Extract the parameters, their values and their constraints on the cases.
- (2) Compare the parameters, their values and their constraints in the problem with those of the case.
- (3) If the constraints are satisfied, start a replay of the case solution with the problem parameters. Replaying means applying the previous planning steps as long as possible.

In this way one investigates whether the data in the problem satisfy the major constraints in the solution derivation.

A special view is taken by activity measures. They consider how many activities two workflows have in common. This can be computed without knowing which dependencies exist between the activities. The measure can be any of the elementary counting measures.

7.7.1 Similarities for Time Series

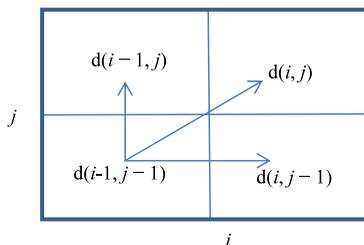
In the dynamic problems, where time plays a role in the easiest situations, the graph structure is very simple. It is totally linearly ordered, representing time. The case base consists of a set of such sequences and for a given query sequence and one searches for the nearest neighbour sequence.

There are many quite different applications, for instance, in economics and the environment. That means the case base contains a set of observed time sequences. A frequent application is prediction. To use CBR it is necessary to compare such sequences.

A simple way of comparing two temporal sequences is Dynamic Time Warping (DTW), a distance technique. It is a dynamic technique that finds a mapping between the measurements from the two time series so that the cumulative value of a given distance function d becomes minimal. For simplification we require that each measurement of one time series is mapped onto at least one measurement of the other time series and that the mapping preserves the chronological order of the measurements.

One denotes the points in the first time series by i and the points of the second time series by j .

Therefore, the algorithm starts by mapping the first elements of each time series and then successively adds new mappings until the all elements have been included. The function $d(i, j)$ denotes the distance between the i th value of the first time series and the j th value of the second time series. It is usually taken as the difference between the costs arising for i and j .

Fig. 7.4 DTW step

If we have a path going through the nodes $(i_1, j_1), \dots, (i_k, j_k), \dots$, then we sum up the differences:

$$D_{path} \sum_k d(i_k, j_k).$$

Then an optimal path gives the distance between two time series. Thus the value $DTW(i, j)$ is calculated as the sum of the distance $d(i, j)$ and the minimal cumulative distance for a mapping that ends with $(i-1, j)$, or $(i-1, j-1)$, or $(i, j-1)$. We illustrate the steps in Fig. 7.4.

Some addition to the time series is of probabilistic nature. That means we deal with stochastic sequences. The difficulty depends on the kinds of probabilities and what is known about them. For instance, the probability models can be unknown and/or dynamic. This makes the whole situation quite complicated. We consider this in Chap. 16, Probabilities.

7.8 Tools

Most general CBR systems offer a variety of measures for complex case representations. We discuss three object-oriented CBR tools.

CBRWorks is a commercial tool, that employs similarity between object-oriented representations (Schulz 1999). Its documentation contains many examples.

CAT-CBR (Abasolo et al. 2002) uses UML for specifying CBR components.

jColibri (Bello-Tomás et al. 2004) is an object-oriented framework implemented in Java. It uses XML to configure its data. The development framework of jColibri supports knowledge reaching from the simplest to the most complex knowledge.

The package DTW implements most known variants of the DTW algorithm family, including a variety of recursion rules (also called step patterns), constraints, and substring matching; see Giorgino (2009).

7.9 Chapter Summary

The similarity measures have been extended to more complex case representations. Here we follow precisely presentations given in Chap. 5, Case Representations, but

some additions are provided. This is intended as a help in choosing an adequate similarity measure for more complex situations. We considered mainly taxonomy-oriented, object-oriented and graph-oriented measures. There are many connections between them. The object-oriented measures are the most complex ones and here we had to consider additional attributes such as relations and many-valued attributes.

The graph-oriented measures are applied to process similarities. A different view was taken on time series when DTW was discussed for comparing time sequences.

7.10 Background Information

For graph similarities, see Bunke and Messmer (1993). A special graph similarity procedure is described in Raymond et al. (2002). An overview can be found in Riesen and Bunke (2009).

For more on group similarity, see Boerner (1993). The Wu–Palmer similarity was introduced in Wu and Palmer (1994).

Object-oriented measures are studied in Bergmann and Stahl (1998).

Process and workflow similarities have been studied in various contexts. For business workflows, see for instance Minor et al. (2007) and Jung and Bae (2006). Replay of action sequences was introduced in Muñoz-Avila and Hüllen (1996). Different measures for workflows are evaluated in Wombacher (2006).

Dynamic Time Warping has been applied to areas like video, audio, graphics and speech recognition (in particular to deal with varying speeds of speech). A general reference for DTW is Myers and Rabiner (1981).

7.11 Exercises

Exercise 1 Define a taxonomic description of your city (or province, or state).

Find a similarity measure that compares parts of the city at different levels of the taxonomy with respect to the social and/or economic structure.

Find a similarity measure that compares parts of the city with respect to house prices.

Exercise 2 Describe cars (or parts of them) using object-oriented representation in two ways:

- (a) For use in configuration
- (b) For failure diagnosis

Motivate the choice for your taxonomic relations. Find for each one a suitable similarity measure.

Exercise 3 Describe a process model for organising a scientific conference and construct an activity similarity. Consider, in particular, mandatory time relations.

Define a similarity measure for these process models.

Exercise 4 Describe two time series based on weather forecasts for two different cities. Compute the similarity between them using a proper metric.

References

- Abasolo C, Plaza E, Arcos JL (2002) Components for case-based reasoning systems. In: Escrig M, Toledo F, Golobardes E (eds) *Topics in artificial intelligence. Lecture notes in artificial intelligence*, vol 2504. Springer, Berlin, pp 1–12
- Bello-Tomás JJ, González-Calero PA, Díaz-Agudo B (2004) JColibri: an object-oriented framework for building CBR systems. In: Funk P, González-Calero PA (eds) *ECCBR 2004: advances in case-based reasoning. 7th European conference, Madrid, Spain, August/September 2004. Lecture notes in computer science (lecture notes in artificial intelligence)*, vol 3155. Springer, Berlin, p 32
- Bergmann R, Stahl A (1998) Similarity measures for object-oriented case representations. In: Smyth B, Cunningham P (eds) *EWCBR-98: advances in case-based reasoning. 4th European workshop on case-based reasoning, Dublin, Ireland, September 1998. Lecture notes in computer science (lecture notes in artificial intelligence)*, vol 1488. Springer, Berlin, p 25
- Boerner K (1993) Structural similarity as guidance in case-based design. In: Wess S, Althoff K-D, Richter MM (eds) *Selected papers. Topics in case-based reasoning. EWCBR-93: first European workshop, Kaiserslautern, Germany, 1–5 November 1993. Lecture notes in artificial intelligence*, vol 837. Springer, Berlin, p 197
- Bunke H, Messmer BT (1993) Similarity measures for structured representations. In: Wess S, Althoff K-D, Richter MM (eds) *Selected papers. Topics in case-based reasoning. EWCBR-93: first European workshop, Kaiserslautern, Germany, 1–5 November 1993. Lecture notes in artificial intelligence*, vol 837. Springer, Berlin, p 106
- Giorgino T (2009) Computing and visualizing dynamic time warping alignments in R: the DTW pack. *J Stat Softw* 31:1–24
- Jung J-Y, Bae J (2006) Workflow clustering method based on process similarity. In: Gavrilova ML, Gervasi O, Kumar V et al (eds) *ICCSA 2006: computational science and its applications. International conference, Part II, Glasgow, UK, 8–11 May 2006. Lecture notes in computer science*, vol 3981. Springer, Berlin, p 379
- Minor M, Tartakovski A, Bergmann R (2007) Representation and structure-based similarity assessment for agile workflows. In: Weber RO, Richter MM (eds) *ICCBR 2007: case-based reasoning research and development. 7th international conference on case-based reasoning, Belfast, UK, August 2007. Lecture notes in computer science (lecture notes in artificial intelligence)*, vol 4626. Springer, Berlin, p 224
- Muñoz-Avila H, Hüllen J (1996) Feature weighting by explaining case-based reasoning planning episodes. In: Smith I, Faltings B (eds) *EWCBR-96: advances in case-based reasoning. Third European workshop, Lausanne, Switzerland, November 1996. Lecture notes in computer science (lecture notes in artificial intelligence)*, vol 1168. Springer, Berlin, p 280
- Myers CS, Rabiner LR (1981) A comparative study of several dynamic time-warping algorithms for connected word recognition. *Bell Syst Tech J* 60(7):1389–1409
- Raymond JW, Gardiner EJ, Willett P (2002) RASCAL: calculation of graph similarity using maximum common edge subgraphs. *Comput J* 45:631–644
- Riesen K, Bunke H (2009) Graph classification by means of Lipschitz embedding. *IEEE Trans Syst Man Cybern, Part B, Cybern* 39(6):1472–1483
- Schulz S (1999) CBRWorks: a state-of-the-art shell for case-based application building. In: *GWCBR'99: 7th German workshop on case-based reasoning, Würzburg, Germany, 3–5 March 1999*
- Wombacher A (2006) Evaluation of technical measures for workflow similarity based on a pilot study. In: Meersman R, Tari Z et al (eds) *OTM 2006: on the move to meaningful Internet systems*

- 2006: CoopIS, DOA, GADA, and ODBASE. Lecture notes in computer science (lecture notes in artificial intelligence), vol 4275. Springer, Berlin, p 255
- Wu Z, Palmer M (1994) Verb semantics and lexical selection. In: ACL'94: 32nd annual meeting of the association for computational linguistics. Association for Computational Linguistics, Stroudsburg, p 133

Chapter 8

Retrieval

8.1 About This Chapter

This chapter assumes you have read the previous chapters. It presents basic retrieval algorithms. It starts with elementary methods such as sequential retrieval. Then it introduces retrieval methods with more complex indexing. It presents retrieval methods that may be useful in certain CBR systems. Readers may select to study some sections only depending on their application interest. More advanced retrieval methods are shown in Chap. 14, Advanced Retrieval. The reading of all previous chapters is recommended.

8.2 General Aspects

Retrieval methods have a prominent position in the CBR cycle. Retrieval in case bases is entirely different from database retrieval. There are major differences. Basically, the problem to be solved is usually not stored in the database. We discuss these aspects in Chap. 23, Relations and Comparisons with Other Techniques.

In CBR, retrieval is not a standalone process. It is strongly connected with the case representation and the similarity measure. In Chap. 2, Basic CBR Elements, concepts we distinguished:

- (a) Attribute-based representations.
- (b) Textual-based representations.
- (c) Dialogue-oriented representations.
- (d) Database representations.
- (e) Image and speech representations.

All these have different retrieval methods because of their different representation forms. One distinguishes two kinds of data retrieval:

- (1) Retrieval algorithms: They operate mainly on a fixed database.
- (2) Agent-based retrieval: This is mainly for search processes in many, not necessarily fixed, databases. This search is often distributed.

In a CBR system one is not searching for some object that is somewhere in the case base; but rather, one presents a query object and searches for the nearest neighbour in the set of answer objects. Here one faces two problems:

- (1) The intended object may not be in the case base, as desired;
- (2) The query may not describe the intended object precisely.

In information retrieval and database systems the retrieval is started by representing a keyword. The problem with this is that one gets either no answer (silence) or too many answers (noise). As a consequence, just naming a key will not be successful. The cases still have to be indexed and the indexing problem is a central task in CBR.

On the positive side, CBR systems do not have silence because one always gets an answer (which may, however, not be very useful) and noise is avoided because the number of retrieved nearest neighbours can be controlled.

8.3 The Retrieval Task

A description of the CBR retrieval task is: We are given a base $CB = \{c_1, \dots, c_n\}$ of objects (a case base, a product base, and so on), a similarity measure sim and a query q (new problem, demanded product, and so on).

We want to retrieve

- the object c_i that is most similar to q ,
- or the k most similar objects $\{c_1, \dots, c_k\}$ to q (ordered or unordered),
- or all objects c_i that have to q at least a similarity sim_{\min} .

The three most interesting properties of a retrieval method are correctness, completeness and efficiency.

Because the purpose of a retrieval function is to find, for a given similarity measure sim and a query q , the k -nearest neighbours, the retrieval has to decide for two given answers a and b which is closer to q . In other words (in the terminology of Chaps. 6 and 7 on similarity), it has to find if one of the two relations ($a \geq_q b$), ($b \geq_q a$) does hold. The retrieval function imposes another ordering on the objects based on how cases are retrieved. This ordering does not necessarily coincide with the ordering given by the similarity measure. In this view the retrieval function induces a similarity measure sim_{ret} and it is not a priori guaranteed that it agrees with sim .

This motivates the following definitions. Suppose ret is a retrieval function.

Definition 8.1

- (i) The function ret is correct with respect to sim if for all a, b, q : $(a \geq_{\text{ret}q} b) \Rightarrow (a \geq_{\text{sim}q} b)$.
- (ii) The function ret is complete with respect to sim if for all a, b, q : $(a \geq_{\text{sim}q} b) \Rightarrow (a \geq_{\text{ret}q} b)$.

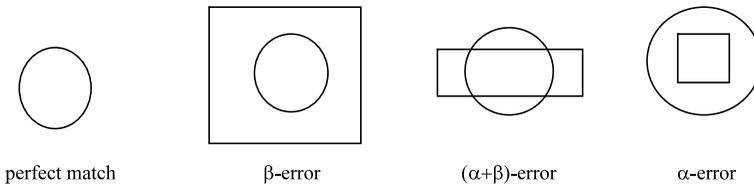


Fig. 8.1 Schematic representation of the errors

(iii) The function ret is complete if it retrieves the nearest neighbours of the similarity measure.

The desire is of course to have completeness and correctness; along these lines, the retrieval function implements the similarity measure. All the methods represented here follow the local-global principle and the search algorithms are implemented along these structures.

Efficiency and accuracy are two important issues for retrieval that are in competition with each other. The efficiency of the different retrieval methods depends very much on the following:

- The representation of the objects.
- The case base structure.
- The similarity measure.
- The accuracy of the intended answer or solution.

These characteristics depend in turn on the domain and the specific properties of the application. The central problem connected with the case base is its organisation and the index structure for accessing it. The index structure is the analogue of the keys in databases.

8.3.1 Retrieval Errors

There are two possible kinds of errors:

- (a) α -error: A case p that is sufficiently similar to q with respect to the underlying measure sim is not selected. This leads to an incomplete retrieval.
- (b) β -error: The methods select a case p that is not sufficiently similar to q . This leads to incorrectness and p has later on to be eliminated. This leads to unnecessary effort and there is only a small performance advantage.

Figure 8.1 shows a schematic representation of the errors. The rectangles describe the retrieved set and the circles describe true nearest neighbours.

These errors will usually have different costs. To some degree one can control them by using a similarity threshold that defines the sufficient and insufficient areas in the sense of rough sets; see the Chaps. 6 and 7 on similarity. Sometimes one introduces the α -error on purpose. This happens if the number of nearest neighbours is undesirably high. We discuss this problem in Sect. 14.3 on diversity.

8.4 Basic Retrieval Methods

When the user has a problem there is still the task of formulating it in such a way that the system can accept it. In various kinds of CBR systems one does not have a choice; for instance, the problem can be an image. In this chapter we restrict ourselves to attribute-based CBR. Other retrieval methods are discussed in Chaps. 17, 18, 19, and 20.

We start with elementary retrieval algorithms. The term *elementary* refers to the property that no complex but rather very simple index structures are used.

8.4.1 Query Generation

Before retrieval can be started a query has to be generated. The query generation depends on the specific situation. There are several possibilities; the major ones are:

1. Automatic generation after a preprocessing step.
2. Interviewing the user:
 - *Form-based* interview.
 - *Dialogue-based* interview.
3. Query-by-example
 - Often in combination with dialogs.

In form-based interviews the user gets a template sheet that can be filled. This is done at once. Dialogue-based interviews will be discussed in detail in Chap. 20, Conversational CBR. Queries can also be represented in spoken form. Speech recognition is discussed in Chap. 19, Sensor Data and Speech.

8.4.2 Filtering

An extreme case of retrieval is that a case is not retrieved under certain circumstances at all because of some strict constraints. If this is known early the search space will be reduced and retrieval efficiency will be increased. The most important situations are when certain values are not allowed for a case.

Example Persons under 18 years of age are not allowed to buy alcoholic beverages, regardless of any other merits.

Such a restriction cannot simply be achieved by assigning the local similarity value zero to an attribute because the case could still be running in the competition for the nearest neighbour. This was discussed in Chap. 6, Basic Similarity Topics.

We observe two ways of doing this:

- (a) By using specific similarity measures.
- (b) By using special retrieving techniques.

Concerning specific similarity measures it has first to be observed that a linear weighted sum measure will not work. One can, however, assign a global similarity value zero to those answers that should be excluded then they will never be considered.

Another possibility is to apply the filtering at retrieval time. We consider this later in Sect. 8.4.4.

8.4.3 Sequential Retrieval

The simplest algorithm is sequential retrieval. It operates on records of objects and gives an array as output. It can be viewed as a brute force approach for finding the k -nearest neighbours.

Next, is the pseudocode description of the algorithm. For simplicity we assume that similarity values of objects are different from each other.

The data structures (which will be used later for kd-trees too) are:

Types:

```

SimObject = RECORD
                object: object;
                similarity: [0..1]
            END;
SimObjectQueue = ARRAY[1..k] OF SimObject;
  
```

Variables:

```

scq: SimObjectQueue      (*m most similar objects *)
cb: ARRAY [1..n] OF object (* object base *)
  
```

The algorithm is simple:

```

scq[1..m].similarity := 0
FOR i := 1 to n DO
    IF sim(q, cb[i]) > scq[m].similarity
    THEN insert cb[i] in scq
RETURN scq
  
```

The complexity of sequential retrieval is linear.

Advantages:

- Simple implementation.
- No index structures have to be built.
- Works for all similarity measures.

Disadvantages:

- Problems for large case bases.
- Retrieval effort is independent from the query.

8.4.4 Two-Level Retrieval

The two-level retrieval acts in two steps when a query q is presented. In the first step some candidates for the nearest neighbour(s) are selected and in the second step the final retrieval takes place.

The point is that one considers certain cases not only because they are not regarded as useful but also because they should be excluded for some reasons despite other merits they may have. This cannot be achieved by setting one or more attribute similarity values to 0 because then these cases will be selected when they should be excluded.

The method for this is called MAC/FAC (Many are called; Few are chosen).

The initial step for the candidates selects $C_q \subseteq CB$ by:

$$C_q := \{p \in CB \mid \text{SIM}(q, p)\}$$

where SIM is a certain predicate.

SIM is not a measure but a binary predicate; therefore, this step is a relational retrieval. The difficulty is in defining SIM in such a way that there are only a few candidates, but not many objects p close to q are excluded. This means that both of the introduced error types are small.

One orders these candidates according to their similarity to q and sequential retrieval takes place.

The MAC stage is a “wide-net” stage in which computationally cheap and structurally simple match processes are carried out. It takes place between the query case in working memory and each of a large corpus of cases stored in long-term memory. A small number of remaining cases, both sound and unsound, is typically obtained.

The FAC stage uses a simple matching in parallel on each of the results of the MAC stage, selecting the best match (or several, if they are sufficiently close to the best).

Examples of the predicate SIM:

Partial equality:

$\text{SIM}(q, p) \Leftrightarrow q$ and p agree in (at least) one attribute. In order to be useful, some knowledge of the importance of the attributes is needed.

Local similarity:

$\text{SIM}(q, p) \Leftrightarrow q$ and p are sufficiently similar for all attributes. This requires a user-defined threshold θ and provides a certain degree of flexibility.

Partial local similarity:

$\text{SIM}(q, p) \Leftrightarrow q$ and p are sufficiently similar for one attribute. Here, combined knowledge about the threshold and the attribute importance is needed.

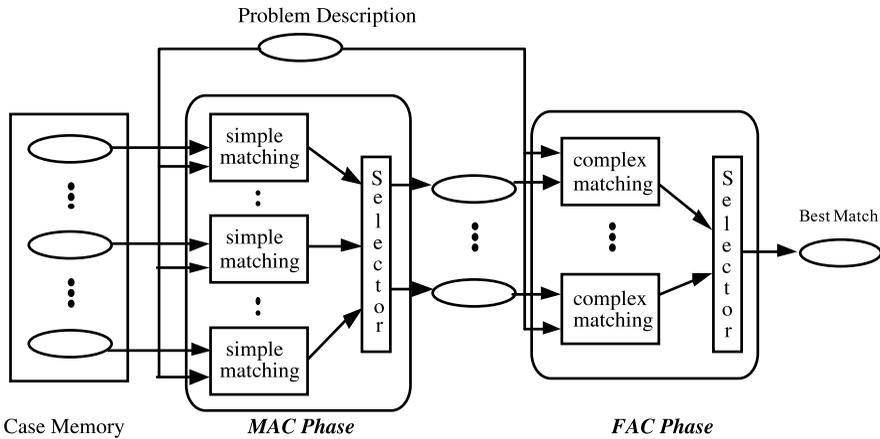


Fig. 8.2 MAC/FAC

The MAC/FAC approach is illustrated in Fig. 8.2.

Advantages:

- There is a performance advantage if in the preprocessing only few cases are selected.

Disadvantages:

- The methods may not select the most useful cases and may violate accuracy.

An important point is that the MAC step can play the role of a filtering. The MAC phase allows such a filtering by not selecting the case.

As a consequence, the MAC step should be taken with much care.

8.4.5 Geometric Methods

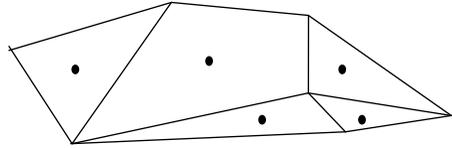
This methods deal with classification problems (or related ones) and offer a geometric view and methods for classes in the n -dimensional real space, where the cases are points in the space and one deals with metric similarities. For simplicity and to illustrate the methods, we consider the two-dimensional situation, i.e., the real plane.

8.4.6 Voronoi Diagrams and k -Nearest Neighbours

This retrieval method requires involved preprocessing methods dealing with the construction of Voronoi diagrams. In fact, almost all of the work is done in the preprocessing.

An ordinary Voronoi diagram is formed by a set of points in the plane called the generating points. Every point in the plane is identified with the generator which is

Fig. 8.3 A simple Voronoi diagram



closest to it by some distance function d . A common choice is to use the Euclidean distance metric, but all the arguments given are valid for weighted distances too.

Suppose a finite set of cases $x \in CB$ of some x is given. G is called the set of generators.

Definition 8.2

- (i) The Voronoi region of $x \in G$ is

$$\text{vor}(x) = \{y \mid \text{for all } z \in G, z \neq x : d(x, y) \leq d(z, y)\}.$$

The boundary of $\text{vor}(x)$ is called a Voronoi polygon.

- (ii) The set of all Voronoi regions is the Voronoi diagram of G .

An example of a Voronoi diagram is shown in Fig. 8.3.

The *Voronoi diagram* has the property that for each site, every point in the region around that site is closer to that site than to any of the other sites.

The boundaries of the polygons, called Voronoi edges, are the sets of locations that can be assigned to more than one generator. The Voronoi polygons that share the same edges are called adjacent polygons and their generators are called adjacent generators. The Voronoi polygon and Voronoi diagram are now formally defined. Assume a set of generators $G = \{g_1, \dots, g_n\}$, $n > 2$, is given.

Definition 8.3 If $d(p, g_i) = \min(d(p, g_j) \mid 1 \leq j \leq n)$ then the region of g_i is

$$V_G(g_i) = \{p \mid d(p, g_i) \leq \min(d(p, g_j) \mid i \neq j)\}.$$

$d(p, g_i)$ specifies the length of the straight line connecting g and g_i in Euclidean space.

The next step is to define a pointer P that identifies for each point p in the plane the Voronoi region of p . Finally, for each generator the list of adjacent generators are stored in a lookup table LOOK. With this construction the preprocessing ends. We observe that in this step some distances are already computed.

Now we take for G the whole case base.

The identification of the nearest neighbour of a query point q now reduces to applying the pointer P to the query q ; the generator of the identified region $V_1(q)$ is the nearest neighbour.

For finding the k -nearest neighbours we look first at the second nearest neighbours.

Observation: The second nearest neighbour of q is one of the generators of the regions adjacent to $V_1(q)$. The reason is that otherwise one had to cross one of the adjacent regions in order reach the second nearest neighbours.

In the two-dimensional case one restricts the number of edges because one can show:

$|\text{edges}| \leq 3|G| - 6$ from which one can conclude that the average number of edges for each polygon is at most 6.

For finding the k -nearest neighbour one has to search the iterated adjacent generators, which is easily realised by searching in the lookup table LOOK.

Advantages:

- Fast search at runtime.
- Easy update if new cases come up.

Disadvantages:

- Preprocessing often too involved to be applicable, in particular, in higher dimensions.
- Restricted to metric similarities and numerically coded case representations.

For higher dimensions the algorithms are much more complex and often impractical.

One can equip the Voronoi diagrams with a weight $\omega(p)$ that is associated with the generating point p . Weights $w > 1$ extend the region while weights $w < 1$ shrink the region. This means cases with higher weights have more influence on their neighbourhood. This leads to a new distance function:

$$d_\omega(q, p) = f(d(q, p), \omega(p)).$$

Here, d is the Euclidean distance and f is a function that increases in the first and decreases in the second argument. With this distance one can define diagrams.

Examples for f are:

- $f(x, y) = x - \omega(y)$
- $f(x, y) = x/y$.

This can be useful if the importance of certain cases is known a priori.

8.4.7 Geometric Approximation

For approximation tasks in the real space one most naturally uses circles. However, numerical computations with circles take up much time. For that reason one replaces the circles by squares, rectangles or more general polygons such as (hyper-) rhombi. The situation is illustrated in Fig. 8.4.

It is easy to introduce weights and it can be done in a way similar to that for elliptic queries shown in Chap. 6, Basic Similarity Topics. Instead of a square or a rectangle, a rhombus is used to show how to represent weights where the y -axis is more important than the x -axis. The most similar cases are within a hyper-rhombus

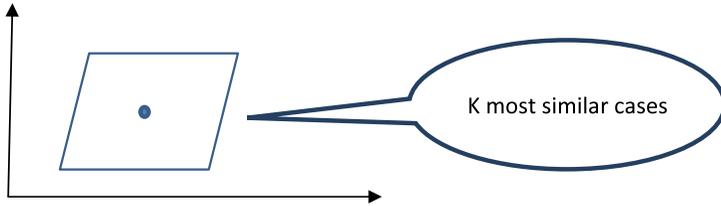


Fig. 8.4 A hyper-rhombus replaces a circle for easier computation

Fig. 8.5 Approximation

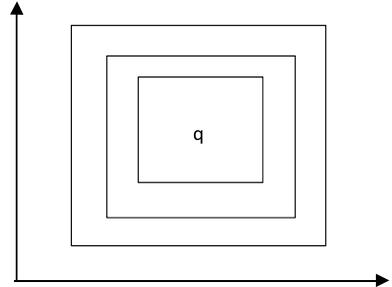
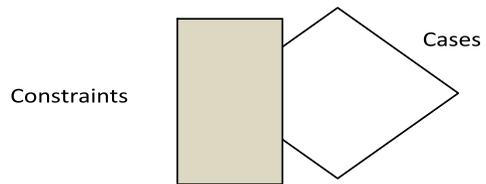


Fig. 8.6 Example of geometrically represented constraints



centred on the query q . Less similar cases are outside the rhombus. For a rectangle, cases within the rectangle have the minimum similarity. One performs the approximation by changing the size of the rectangle, as shown in Fig. 8.5.

There are two reasons to change the size of the rectangle:

- If one needs more nearest neighbours it can be extended.
- If one wants a higher similarity it can be shrunk.

Advantages:

- Efficient and easy to implement.

Disadvantages:

- Restricted case representations and similarity measures.

8.4.8 Geometric Filtering

Filtering out unwanted cases is easy if the constraints describing them are of a simple geometric character. An example is given in Fig. 8.6.

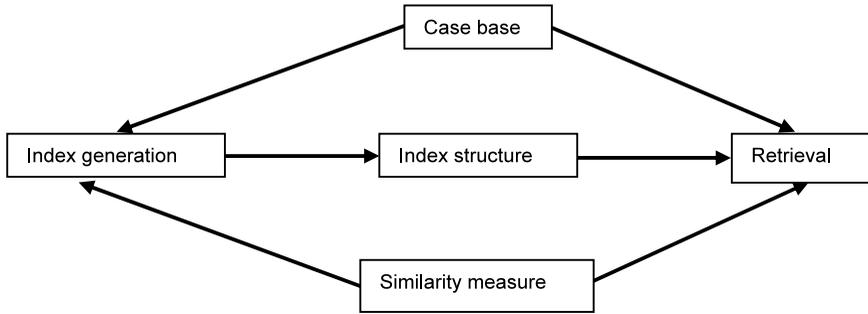


Fig. 8.7 Design steps

In such situations efficient geometric computations are available for cutting out the forbidden areas.

8.4.9 Index-Based Retrieval

Indexing cases is one of the most challenging tasks in CBR. The index structure is supposed to guide the search for the most similar case(s). Therefore, two steps have to be performed:

- (1) Generating the index structure; this is a preprocessing step.
- (2) The retrieval itself.

The basic steps are shown in Fig. 8.7.

One can distinguish between indexed and unindexed attributes. Indexed attributes are used for retrieval and are predictive for finding the case's solution. Unindexed attributes are not used for retrieval and not predictive of the case's solution, but may contain useful information when the case is retrieved. See the following example:

Attribute: Hotel location.

Indexed attribute: Close to the beach.

Unindexed attribute: 2915, Main Street, Mailbox M3CB

The index generation for the whole case base takes place off-line while the retrieval is online. The more complex the case structure is, the more the effort the index generation will require. Although one cannot formally define what a good indexing is, there are several guidelines for it. First, indices should be predictive and discriminatory.

What should (could) be indexed?

- Problem properties.
- Solutions properties.
- Processes.
- Context properties.

Index selection methods include:

- Predefined indices.
- Hierarchies.
- Balanced, statistical criteria.
- Biased, context-dependent criteria.

8.4.10 *kd-Trees*

Trees are typical examples of index structures. In database retrieval the use of trees is common. The attributes are already fixed when the tree is generated. The basic idea is searching top-down and eliminating all other branches when taking a specific branch. This is not directly applicable in CBR. The reason is that we are not searching for a specific object but rather for one that is most similar to the query. Nevertheless, the idea of decomposing the data (i.e., case base) iteratively into smaller parts and using a tree structure is still an advantage. In contrast to the database situation, backtracking in the tree is sometimes necessary when the wanted object is not found. Nevertheless, it is advisable to try using trees for retrieval purposes in CBR too.

Although there are other possibilities, in CBR the most used trees are the kd-trees. The “kd” stands for k -dimensional. These trees are the index structures. The basic purpose of the trees is to represent the cases in a homogeneous way. This method does not universally work even for attribute-value representations and some restrictions are required. They state that the domains of the attributes are ordered and the similarity ordering is compatible with the ordering on the attribute domains, i.e.:

If $x_i <_i x_{i'}$ then

$$\text{sim}((x_1, \dots, x_n), (x_1, \dots, x_{i'}, \dots, x_n)) \geq \text{sim}((x_1, \dots, x_n), (x_1, \dots, x_i, \dots, x_n)).$$

The kd-tree is the index structure that one constructs in the preprocessing step.

Suppose there are k ordered domains T_1, \dots, T_k for the attributes A_1, \dots, A_k and a case base $CB \subseteq T_1 \times \dots \times T_k$ is given. Furthermore, there is a user-defined parameter b for the sizes of the buckets. The buckets are sets that will be searched (at the end of the procedure) sequentially.

The parameter b represents the number of cases for which the effort for linear retrieval is acceptable (linear retrieval within a bucket). Hence, b should be set with respect to the computational effort of the similarity calculation and to the respective notion of “acceptable”.

The index structure will now be a kd-tree which is a recursively defined annotated tree.

Definition 8.4 A kd-tree $\text{Tree}(CB)$ for the case base CB is a recursively defined binary tree:

- (i) If $|CB| \leq b$ then $\text{Tree}(CB)$ is a leaf node (called bucket).
- (ii) If $|CB| > b$ $\text{Tree}(CB)$ is a tree where the root is annotated with some attribute A_i and some value $v_i \in T_i$.

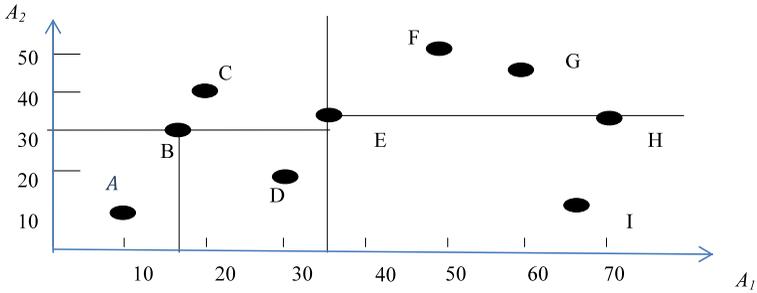


Fig. 8.8 Geometric view

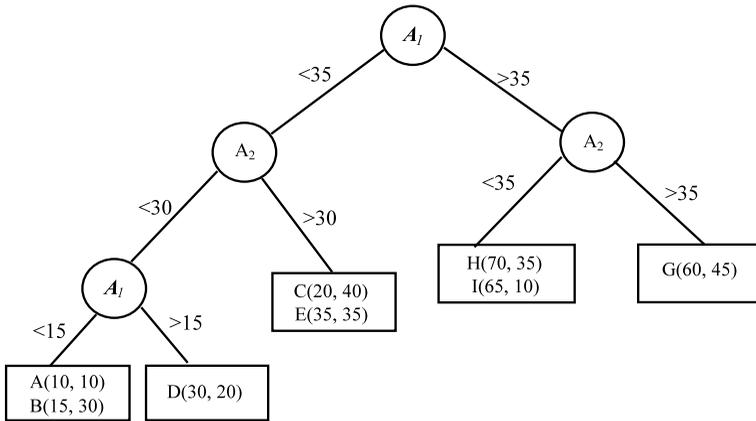


Fig. 8.9 Tree view

The root has two kd-trees as successor trees:

$Tree(CB_{\leq})$ and $Tree(CB_{>})$ where $CB_{\leq} := \{x \in CB | x \leq v_i\}$ and $CB_{>} := \{x \in CB | x > v_i\}$.

A kd-tree partitions a case base iteratively; the buckets at the leaf nodes are not partitioned further. In the sense of the local-global principle the tree provides the global structure.

We show an example of a kd-tree with a case base $CB = \{A, B, C, D, E, F, G, H, I\}$ and assume two real-valued attributes A_1 and A_2 ; furthermore the bucket size will be 2. This allows two visualisations; a geometric one and a tree-oriented one. The geometric view in Fig. 8.8 shows just rectangles:

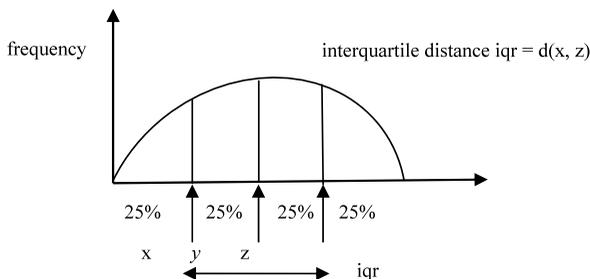
The corresponding tree view is shown in Fig. 8.9.

The generation of the kd-trees follows the recursive definition. The generating algorithm is:

```

PROCEDURE CreateTree(CB): kd-tree
  IF |CB| ≤ b
    THEN RETURN leave node with base CB
    
```

Fig. 8.10 Interquartile distance



ELSE

$A_i := \text{choose_attribute}(CB);$

$v_i := \text{choose_value}(CB, A_i);$

RETURN

Tree with the root labelled with A_i and v_i that has the two subtrees

CreateTree($\{(x_1, \dots, x_k) \in CB | x_i \leq v_i\}$) and

CreateTree($\{(x_1, \dots, x_k) \in CB | x_i > v_i\}$).

At each node one has two degrees of freedom, the choice of the attribute and the choice of the value that partitions the domain. We give some examples of both that rely on statistical knowledge of efficient search. The goal is to minimize the length of the path to the buckets. For this, one chooses the inner nodes in such a way to obtain a tree that is most likely balanced. Very different cases are separated very early (which means high in the tree).

A typical method for attribute selection uses for real-valued T_i the one that has the largest interquartile distance, as illustrated in Fig. 8.10.

The first quartile x (25 % quartile) divides the lower half of the distribution into two equally sized areas while the third quartile z (75 % quartile) does this with the upper half of the distribution. The greater the distance between these quartiles, the greater the dispersion of the attribute values.

During tree construction the attribute having the maximal dispersion with respect to the local similarity measure used, sim , is selected as the discriminating one. It denotes that we select an attribute for discriminating purposes where the respective quartiles have the lowest local similarity (which corresponds to the greatest distance).

For defining the splitting point there are mainly two methods: Median splitting and maximum splitting.

- (1) Median splitting: Choose the median d as partition point; see Fig. 8.11. This has the advantage of a uniform splitting of the cases.
- (2) Maximum splitting: Choose the largest gap as in Fig. 8.12.

The picture of the rectangles is now disturbed by the similarity measure. The point one is looking for may not be on the tree but rather (in the real-valued geometric view) somewhere in the area and be located in a part that is no longer considered.

Fig. 8.11 Median splitting

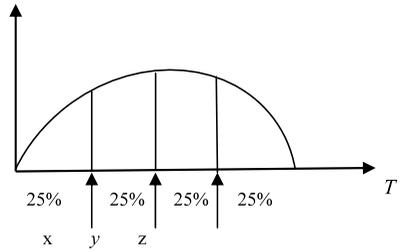


Fig. 8.12 Largest gap splitting



It may have a smaller distance to the query than other points among those still under consideration. The tree considers hyper-rectangles while the similarity considers hyper-balls. For retrieval purposes the two have to be put into a relation. We consider the two situations shown in Fig. 8.13(a) and (b).

The left side is easier to handle, the right side may cause backtracking. To find out which situation is present one needs two tests:

- (1) The BWB test: Is the ball within the bounds?
- (2) The BOB test: Does the ball overlap the bounds?

For the BWB test we use the computationally easy bounding box shown in Fig. 8.14.

If the bounding box is included in the bucket the test is passed. Otherwise the search has to be continued.

The BWB test is not a decision procedure but in the positive case one is sure that there is no case k in the neighbouring subtree that is more similar to the query than the found k most similar case. In this test the ball around the query is replaced by a rectangle. This ball is defined by the present k -nearest neighbour. The rectangle is obtained by adding to each coordinate of the query in each direction the radius of the ball.

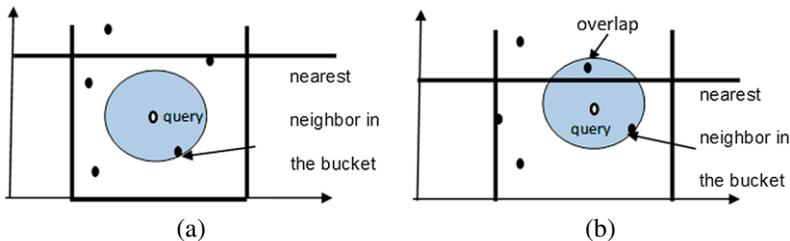
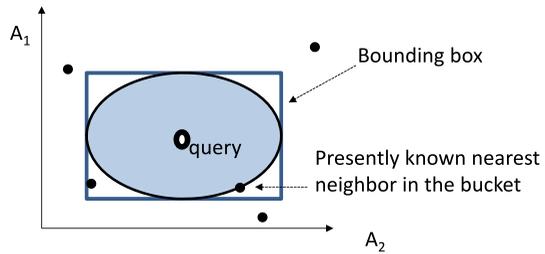


Fig. 8.13 (a) BWB: ball within bounds, (b) BOB: ball overlap bounds

Fig. 8.14 Passed test

For the BOB test we use the same method by investigating the bounding box. If, during a BOB test, a more similar case is found within a neighbouring bucket, then the size of the ball around the query is decreased accordingly.

The tests are sometimes unnecessarily slow. Often the whole space is sparsely populated with cases and in an overlap situation there may be no cases in the overlap at all but the algorithm is still searching for them. It is the purpose of dynamic bound tests to improve the situation. This discussion is beyond the intended complexity of this text. With the two tests we are able to formulate the retrieval algorithm.

In the next algorithm one collects the cases in a record together with their similarities to the query q . We denote the value of the attribute A_i in the query q as $q[A_i]$. Then scq denotes a gain in the actual ordered sequence of nearest neighbours, together with their similarity to the query; it is continuously updated. A sketch of the algorithm is:

- The algorithm searches recursively in subtrees.
- For an inner node the right or left subtree is called depending on the value of the query attribute.
- It is tested by the BOB test if it is necessary to search also in the other subtree.
- For a leaf node the BWB test checks if the search terminates or backtracking has to take place.

In a pseudocode this can be described as the following algorithm:

```

Procedure retrieve (K: kd-tree)
Parameters: Case Base  $CB$ , Integer  $b$ ,  $K$  a node.
IF  $K$  is leaf node (*bucket)
THEN
  FOR each object  $C$  of  $K$  DO
    IF  $\text{sim}(Q, C) > scq[m]$ 
    THEN insert  $C$  in  $scq$ 
    ELSE (* inner node *)
      IF  $A_i$  is the attribute and  $v_i$  the value that labels  $K$ 
      THEN IF  $Q[A_i] \leq v_i$ 
        THEN retrieve( $K_{\leq}$ )
        IF BOB test is satisfied
        THEN retrieve( $K_{>}$ )
        ELSE retrieve( $K_{>}$ )

```

```

IF BOB test is satisfied
THEN retrieve( $K_{\leq}$ )
IF the BWB-Test is satisfied
THEN terminate retrieval with scq
ELSE RETURN

```

Despite the effort, BWB is useful as it may allow for significant gains in retrieval efficiency. Normally, a BOB test is required at each recursive stage of the kd-tree retrieval algorithm. If the BWB test returns true, the recursion terminates (and no further BOB test will be required).

The main properties of retrieval with kd-trees are now summarized. In contrast to the database situation, backtracking in the tree is sometimes necessary when the desired object is not found. In the best situation, there is no backtracking required. This leads to a computation time that is proportional to the depth of the tree; the retrieval effort is $O(\log(n))$, with n being the number of cases in the case base. However, in the worst case, in which backtracking is required for every node, the whole tree must be investigated again, leading to an additional retrieval effort of $O(n)$.

Unfortunately, the situations with few backtrackings, which seldom occur in practice, lead to a high degree of inefficiency. This restricts its applicability in practice.

Advantages:

- Effort dependent on the number m of objects to find.
- Incremental extension possible if new objects arise.
- Storage of the objects in a database possible.

Disadvantages:

- High effort to build up the index structure.
- Inefficient when backtracking is needed.
- Restrictions for kd-trees.
- Ordered domains only.
- Problems with unknown and uncertain values: How do we traverse the tree if values are not known or uncertain?
- Only monotonic similarity measures compatible with the ordering allowed.

8.4.11 Integration with Decision Trees

Classical decision trees allow efficient and correct retrieval methods. They and kd-trees have much in common. A difference is in the annotation of attributes to the nodes. In addition, decision trees allow us, as databases do, to retrieve only cases that are stored in the tree. The idea behind an integration is to combine the advantages of kd-tree retrieval and decision tree retrieval. The starting point is to represent the cases in a tree without information on what kind of retrieval will be used. This applies to problems where decisions, like classifications, have to be made and the

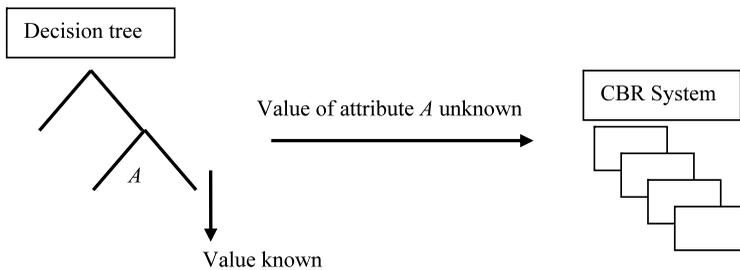


Fig. 8.15 When the kd-tree is called

Table 8.1 Comparison between kd-trees and decision trees

Aspect	Decision tree	kd tree
Similarity measure	Not necessary	Necessary
NN classification	Not possible	Done
Result	Decision, classification	Nearest neighbours
Backtracking	Not necessary	Necessary
Cases	Not necessary	Necessary
Unknown values	Difficult	Possible

representation uses numerical values as in kd-trees. Decision trees were introduced in Chap. 5, Case Representations. When used for retrieval, they can be very fast. There are, however, a number of drawbacks and problems connected with decision trees. The first is that unknown or inexact values for the attributes are problematic to handle.

The idea of a combined use of trees and case bases is illustrated in Fig. 8.15. One starts with the decision tree and arrives at an attribute A . The value of A is needed then.

This means one regards the tree first as a decision tree. If the case is stored in the tree one will get a precise answer. If this does not work then one regards the tree as a kd-tree and switches to the CBR system.

The question is, how can this work? The answer is that one should not change the case representation between the two methods. Because the representation is primary and is given in a tree form, the answer is to use kd-trees for the case representation. However, in kd-trees one has problems with symbolic values. The comparison between kd-trees and decision trees is illustrated in Table 8.1.

When combining decision trees and CBR retrieval in a kd-tree we first look into areas of problems covered by the trees as indicated in Fig. 8.16. If no suitable case is found then CBR retrieval is applied.

I, II, III and IV denote the regions obtained by the tree. The grey-shaded rectangles denote the areas where the solutions are obtained by retrieval from the tree. The circle denotes similarity-based retrieval. For a complete integration there remain still many problems to be solved.

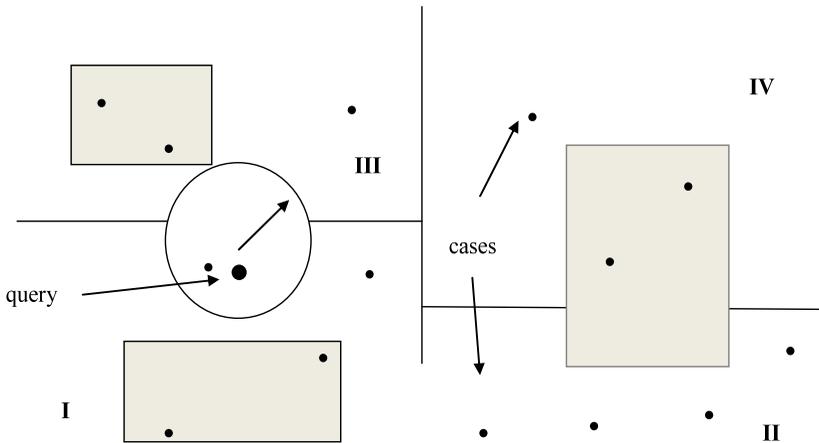


Fig. 8.16 Geometric areas where decision trees are called

8.5 Tools

An efficient retrieval tool mainly for e-commerce applications is Orange; see Schumacher (2002). The CBR Shell Java V1.0 provides more than one approach for retrieval (<http://www.aiai.ed.ac.uk/project/cbr/cbrtools.html>).

8.6 Chapter Summary

Section 8.4, Basic Retrieval Methods, contains, among others, sequential retrieval, two-level retrieval, and index-based retrieval. In addition, we discuss geometric methods and Voronoi diagrams.

Index-based retrieval needs two steps, one for generating the index structure and one for the proper retrieval. As a major example of index-based retrieval, we consider kd-trees. This is reminiscent of databases, but with the additional task of needing searching for cases that are not stored. For this purpose two tests have been presented, the BOB test and the BWB test.

For an improvement of kd-trees the combination with decision trees was discussed.

8.7 Background Information

MAC/FAC has been successfully used to model the results of psychological experiments on similarity-based retrieval; see Gentner and Forbus (1991). Voronoi diagrams are over a century old but their use for finding nearest neighbours is more

recent. An overview over such applications of Voronoi diagrams can be found in Okabe et al. (2000). The method can be generalized to graph representations with the edit distances; see Kolahdouzan and Shahabi (2004). Advanced retrieval methods for geometric representation are given in Stottler et al. (1989). kd-trees are known for decades and used in databases; see Bentley (1975). The use for CBR purposes goes back to Wess et al. (1993). The integration of decision trees and CBR was realised in the INRECA system. The combination of kd-trees and decision trees is an integration of symbolic learning and nearest neighbour classification, improving both of them. This means that cases can be compiled (in the tree) or interpreted in the CBR style. This is described in detail in Auriol et al. (1995), Althoff (1997), and Bergmann (2001).

8.8 Exercises

Exercise 1 Suppose a probability distribution for the attribute-values in the case base is given. Define on this information a predicate SIM for selecting the cases in the first phase.

Exercise 2 Prove that the average number of Voronoi edges per Voronoi polygon (in two dimensions) is at most 6.

Exercise 3 (kd-trees)

- (a) When setting the value of the bucket size b , what must be taken into consideration with respect to the retrieval?
- (b) If m denotes the number of nearest neighbours to be retrieved, which relation between m and b should hold?
- (c) The BWB test and BOB tests are, if repeated and if many attributes are present, computationally expensive. Under what conditions is the effort for doing these tests justified?

References

- Althoff K-D (1997) Evaluating case-based reasoning systems: the INRECA case study. Habilitationsschrift, University of Kaiserslautern
- Auriol E, Wess S, Manago M et al (1995) INRECA: a seamlessly integrated system based on inductive inference and case-based reasoning. In: Veloso MM, Aamodt A (eds) ICCBR-95: case-based reasoning research and development. First international conference on case-based reasoning, Sesimbra, Portugal, October 1995. Lecture notes in computer science (lecture notes in artificial intelligence), vol 1010. Springer, Berlin, p 371
- Bentley JL (1975) Multidimensional binary search trees used for associative searching. *Commun ACM* 18:509–517
- Bergmann R (2001) Highlights of the European INRECA projects. In: Aha DW, Watson ID (eds) ICCBR 2001: case-based reasoning research and development. 4th international conference on case-based reasoning, Vancouver, BC, Canada, July/August 2001. Lecture notes in computer science (lecture notes in artificial intelligence), vol 2080. Springer, Berlin, p 1

- Gentner D, Forbus KD (1991) MAC/FAC: a model of similarity-based retrieval. In: Proceedings of the cognitive science society. Erlbaum, Hillsdale, p 504
- Kolahdouzan M, Shahabi C (2004) Voronoi-based k-nearest neighbour search for spatial network databases. In: Nascimento MA, Özsu MT, Kossmann D et al (eds) VLDB 2004: thirtieth international conference on very large databases, Toronto, Canada, August 31–September 3 2004. Morgan Kaufmann, San Mateo, p 840
- Okabe A, Boots B, Sugihara K, Chiu SN (2000) Spatial tessellations: concepts and applications of Voronoi diagrams, 2nd edn. Wiley, England
- Schumacher J (2002) Empolis Orange—an open platform for knowledge management applications. In: Minor M, Staab S (eds) Experience management: sharing experiences about sharing the experience. Papers from the 1st German workshop on experience management, Berlin, 7–8 March 2002. GI, Bonn, p 61
- Stottler RH, Henke AL, King JA (1989) Rapid retrieval algorithms for case-based reasoning. In: IJCAI 1989: eleventh international joint conference on artificial intelligence, vol 1. Morgan Kaufmann, San Mateo, p 233
- Wess S, Althoff K-D, Derwand G (1993) Using kd-trees to improve the retrieval step in case-based reasoning. In: Wess S, Althoff K-D, Richter MM (eds) Topics in case-based reasoning. EWCBR-93: first European Workshop, Kaiserslautern, Germany, 1–5 November 1993. Lecture notes in artificial intelligence, vol 837. Springer, Berlin, p 167

Chapter 9

Adaptation

9.1 About This Chapter

This chapter is addressed to readers interested in adaptation of the query or the solution. Not all applications need that but it is of relevance to many. The goal of this chapter is to present the main principles and methods for adapting cases. It is assumed you have read the chapters in Part II.

9.2 General Aspects

This chapter is devoted to the reuse part of the CBR cycle. A new problem is never exactly the same as any previously recorded solved problem and one cannot expect that one of the former solutions exactly works for the new problem. That does not mean that the old solutions are useless. Humans know this if they use an experience from the past: It always gives an idea on how to solve a new problem and mostly the solution is an adaptation of a previous solution. There are different ways to make use of this. We will be discussing them in this chapter. The approach is called reuse. A major point is to do it systematically so that one can at least partially automate them. If the solution is not satisfactory, one can also try to change the query slightly. We discuss this systematically in Chap. 20, Conversational CBR, where we discuss how the CBR system can slightly adapt the query without a dialogue.

Reuse of solutions generally requires a *change* of previous solutions, which in turn requires actions. One calls this change *adaptation*. To be systematic, one describes actions that cause changes usually by rules that say under which conditions which adaptations are possible or recommended. Because the ultimate goal is to use such changes for getting good solutions, this chapter is closely connected to retrieval. The problem with retrieval here is that one is searching for the best solution after adaptation. In this situation simple nearest neighbour search in the case base is insufficient. Therefore, one has to extend the methods in Chap. 8, Retrieval.

Adaptation of solutions can either use the previous solution and transform it or use the previous solution strategy to generate a new adapted solution or a part of it. Both are investigated in this chapter.

A simple situation is car repair when the experience is concerned with changing the left tire and the new problem is concerned with the same observations except that one is dealing with the right tire. Then the adaptation is obvious.

For reusing a solution strategy we will first provide some examples. In a case-based system designed to prescribe an exercise plan for sport activities a previous solution consisting of running can be transformed into one of swimming, as shown in Chap. 2, Basic CBR Elements. This could be done through an adaptation knowledge base that would determine that running should be replaced with swimming for individuals presenting knee problems as an attribute.

To reuse a solution for generating a new one, consider a strategy you developed to play a game. When playing the same game later under similar conditions, you may apply the same strategy.

To reuse a solution to generate a new one, consider a strategy used to find a parking place outside the university library building. The strategy is to position your car at the parking lot as close as possible to the building entrance and follow students who are leaving the building to get their cars, leaving a vacant spot for you. This is the kind of strategy to generate new solutions that reuses only the strategy from the old solution.

All these adaptations are performed by actions. Actions have been introduced in Chap. 5, Case Representations, often in the context of plans. They change a given model. If performed in reality they change the world. Therefore, adaptations are special kinds of actions, described formally by rules. These descriptions include input and output types, possible side effects as well as the conditions under which an action can be performed. Because usually more than one action is needed for adapting a solution it is no surprise that adaptation plans enter the stage.

Now we come to a new aspect. When, for instance, a seller is confronted with a demand that is difficult or not at all possible to satisfy the seller has two options. The first one is to adapt its product as just outlined. The second one is to convince the buyer of another alternative as, for instance, “why do you prefer a grey car? We have a white one and this colour is much more fashionable”, which means one does not adapt the solution but rather the query.

This happens not only in e-commerce but in all of CBR. We observe:

- A query can be unsatisfactory.
- A solution can be unsatisfactory.

The term unsatisfactory has in both cases very different meanings that we will discuss later.

Actions can change queries as well as solutions. Query changes complete or correct unwanted, or incompletely or inconsistently formulated queries. This is a topic in Chap. 20, Conversational CBR. Here we discuss some simple methods that one can automate.

Summarizing these observations one finds that the need for adaptation comes from three facts:

- Often there is a lack of cases. In constructive problem solving all designs are unlikely to be represented in the case base. Retrieved cases suggest an initial “design” and adaptation alters the “design” to reflect novel feature values. In electronic commerce an available product may not match exactly the demand and may have to be adapted. One has to find out if this is possible.
- CBR provides approximate solutions only. When using an experience the old situation in the case is not the same as the actual one, and one has to modify it.
- Queries may be insufficiently formulated.

As an example of complexity problems, we consider in e-commerce the selling of PC’s, for which we encounter huge spaces for both problems and solutions:

Problem Space: The set of possible combinations of requirements are numerous, for example:

$$P = 2^{\{\text{Text processing, Games, Music, ...}\}}$$

Solution Space: The set of possible PC configurations is again very large:

$$S = 2^{\{\text{xyz mainboard, 4.3 GB hard disc, ...}\}}$$

Clearly, this space is far too large for exhaustive searching. In fact, often only a relatively small set of queries and solutions can be stored as cases in the case base or as products in a catalogue. We call the space of all products the catalogue space.

In the PC example, the huge set S can be listed compactly by giving one example and a list of possible mainboards, hard discs, and so on possibly with certain constraints.

In general, it will be much larger than the case base because one can obtain the majority of products by adaptation from the few ones stored. The explicitly stored cases may not be sufficiently useful and one can view the adaptation as an improvement of their quality. We discuss this in Sect. 9.8 on quality issues.

In the CBR cycle, adaptation takes place in the reuse phase. In principle, adaptation can vary between null adaptation and the construction of a completely new solution. The latter alternative has no place in CBR. The former occurs quite often. This happens, for instance, when a solution is not adaptable.

9.3 Rules

For a systematic use of adaptation one needs a rigorous formalism. For this purpose, we define adaptation steps by rules as actions in general. We start with topics that are useful for the standard and most often occurring applications. At the end we discuss applications that are more special as well as some general insights. A rule is of the form

$$\phi_1 \wedge \phi_2 \wedge \dots \wedge \phi_n \Rightarrow \text{Action.}$$

The ϕ_i are preconditions that specify the constraints for performing the action.

Although the representations can principally be of any kind we restrict the discussion in this chapter to attribute-based representations (flat or complex). They have to be satisfied before the action can be performed. In other words, the preconditions are constraints that restrict the usage of actions. The single preconditions are combined by logical conjunctions.

9.3.1 Preconditions

The precondition of a rule consists of conjunctions of conditions, i.e., all conditions of the precondition part of the rule must be fulfilled to fire the rule. In other words, they are the so-called hard constraints.

In general, rules contain variables that are local for the rules. That means rules use the attributes in the case description and refer to their values. If in an attribute the value is not specified, the symbol “?” or, better, a variable x can be used as the value. One has to specify what happens then with the action.

The process of checking the satisfaction of a precondition checks it for the case under consideration. When applied to the case, the variables are instantiated with the value the variable has in the considered case. That means preconditions of a rule must be defined in terms of variables accessible to the case description.

If, for instance, the case description contains an attribute A_{size} and one of the preconditions for the variable size is $size > 1000$ then:

- For Case₁ with $A_{size} = 1200$ the condition is satisfied.
- For Case₂ with $A_{size} = 800$ the condition is not satisfied.

Variables may also occur in several preconditions of a rule as a means of sharing values between different conditions contained in the preconditions of the rule. Because the variables can also occur in the action, they are also instantiated there with the same values as in the preconditions.

Variables can hold any kind of basic value or object, or can hold just the class name of an object.

9.3.2 Actions

Actions have been introduced in Chap. 5, Case Representations. The difference is that plans can be of very general character while adaptations deal with cases. Cases in a base have a representation form that is obligatory for all cases and therefore the adaptation actions are more uniform.

Formally, actions have an operator description as mappings:

Operator: states \rightarrow states.

State is a description of something of interest. In CBR it will be a case description of problems as well as solutions. Often, a state description will not describe the whole model or case under consideration but rather a part of it that is considered as relevant. The two most important representation formalisms for our purposes are:

- (1) Stating values of certain attributes as in $A_{size} = 800$ or $Colour(car) = white$.
- (2) Stating the validity of certain predicates as in $brother(john, bill)$.

Strictly speaking, an action generates a case (which may already exist). Sometimes, it is possible that this does not happen in reality. Suppose for instance one wants to change a product. Then one has to ensure that the changed product is available. One can however also change a query as presented below.

Basically, there are only three major types of actions, as mentioned before:

- Adding or creating something.
- Deleting something.
- Modifying something.

Modification can be seen as a macro of the two other actions. The “something” is usually:

- The value of a variable.
- A part of an object description, for instance, a subclass.

Applications of operators can be iterated and arranged in a sequence:

$$state_1 \rightarrow state_2 \rightarrow \dots \rightarrow state_n$$

where the employed actions in the sequence can be (and usually are) different.

9.3.3 Types of Rules

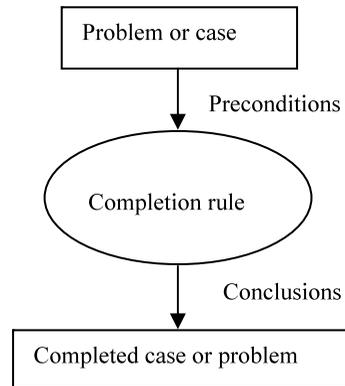
There are several different kinds of rules that will be discussed now.

Mainly, we distinguish two kinds of rules according to the intended actions:

- Completion rules: They adapt case descriptions, in particular, in the query.
- Adaptation rules: They adapt the solution.

9.3.3.1 Completion Rules

When queries are represented they may not be perfect. It is the task of completion rules to investigate and correct this and to avoid conversation problems. One intention is that the rules are completing and correcting the description of a problem or a query. In an incomplete query certain variables may not be instantiated. It may be that certain essential values that cannot be derived from other ones that are missing.

Fig. 9.1 Completion rules

Another “accident” is if the query is inconsistent. The general view on completion rules is shown in Fig. 9.1.

The preconditions of a rule should be defined in terms of variables accessible to the case. A main purpose of the completion rules is to compute the values of certain attributes on the basis of known values of other attributes. The major applications are as follows:

- (a) To compute values of attributes with major importance (i.e., weight) and simplify the similarity computation. In Chap. 6, Basic Similarity Topics, we introduced them as virtual attributes.
- (b) The rules compute values of attributes that are not used for reasoning but are of interest to the user. These are often the so-called metadata. Moreover, one can also specify an arbitrary function of interest to the user, which calculates a new value using the given values.
- (c) There may be missing values in a query. These values have to be created and filled in. This is further discussed in Chap. 13, Advanced Similarity Topics.

A simple example of rules dealing with metadata occurs when customers want to buy a house:

$$\text{IF size} = x \text{ in sqm AND price} = y \text{ THEN price_per_sqm} = y/x.$$

Metadata is also a popular form for introducing additional and sometimes necessary new attributes. This occurs often in a statistical context where concepts like mean or variance occur that may be either crucial for a decision of at least be of interest to a user.

For example, suppose there is a list containing the prices of a product for each day of the year. Then computable metadata are, for instance:

- Average prices of each month.
- Tendencies of price development, and so on.

For missing data the problem is that there is no computation rule given. If one requires a specific value of the domain of the attribute, this can be done in various ways, for instance:

- Using a default value.
- Computing an average.
- Using an optimistic or pessimistic value.

A more difficult role of the rules is if they are used as oracles. That means they provide immediately the solutions of difficult subproblems. An example would be a certain very large number n . Then the prime factors of n are p_1, p_2, \dots, p_k .

This uses a fairly complex and lengthy computation that will not explicitly occur in the rule. For the rule it is enough to store the results in an answer table. In the given example it will occur sometimes that an answer is “do not know”. This is, however, typical.

The rules give values for virtual attributes that do not yet have values. The value can be a constant or computed from values of other variables. The rule has to specify the function that computes the values.

An even more involved kind of rule occurs in object-oriented representations. There the values of variables can be of a more general kind.

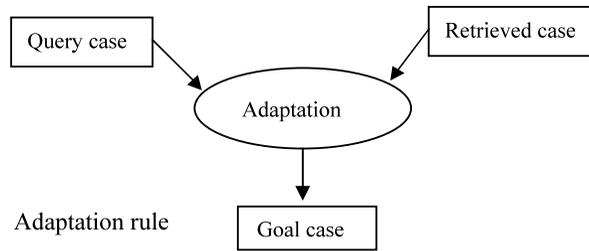
Completion rules can also be used to correct queries. The main application is connected with consistency and the plausibility of the query.

- (1) A demand or query can be consistent or inconsistent. It is inconsistent if the demands contradict each other. Inconsistency is an indication of a knowledge gap, i.e., the query originator (for instance, the customer) does not have knowledge to specify his needs consistently. Inconsistency can be handled by notifying the user about the inconsistency and to requesting him to do something.
- (2) The query is consistent but highly implausible. This can be handled by notifying the user and by suggesting solutions that most likely best fulfil the anticipated user needs (but not necessarily exactly).
- (3) The query is both consistent and plausible but the seller wants to increase profit. This is handled by changing the demand though profit improving rules.

Examples:

- (1) Demands for buying a computer:
 - Memory: 1 GB RAM
 - OS: Windows 7
 These are inconsistent. For detecting this fact a consistency checker is needed.
- (2) Demands for buying a text book:
 - Area: Physics
 - Level: Elementary
 - Author: Nevton
 Although there exists a physicist Nevton, this is not very plausible. Suggestion: Do you mean Newton? Such tests are standard in information retrieval.
- (3) Demands from a car buyer:
 - I want a car for my family with the properties P .

The answer can be that there is a different car that conflicts with your demands but is more useful for your intentions. Such a query adaptation is usually in the interest of the seller. As a precondition, a plausibility checker is needed here.

Fig. 9.2 Adaptation rule

Part of the car description in a diagnostic situation: Brand: Ferrari; hp: 25; max speed: 180 m/h; colour: black; does not start in the morning.

This description needs to be changed in several ways. It has an inconsistency, a redundancy and missing parts.

In general, all completion rules are always active, i.e., they can be applied at any time. However, the sets of rules have to be consistent in the sense that there are no two or more rules giving contradicting values for the same variables in the same case. Such contradictions occurs when one rule says a variable should be set to the value a and another rule says the value should be b , with $a \neq b$. This does not mean there are no rules that set the same variable sometimes to a or to b . In such cases the preconditions have to take care that no contradictions arise.

The problem becomes more difficult if iterated applications of rules take place. Even theoretical methods do not provide sufficient solutions, for instance, if termination criteria are wanted.

9.3.3.2 Adaptation Rules

The actions in the solution adaptation rules are of the form

Adapt: Solutions \rightarrow Solutions.

While the completion rules refer only to two case descriptions, cases adaptation rules refer to three: The query case, the retrieved case and the goal case. The reason is that the query is not answered satisfactorily by the retrieved case, i.e., the goal is not reached. Therefore, one has to compare the three cases and has to adapt the retrieved case as shown in Fig. 9.2.

The adaptation can be applied when the solution is a single object as well as when in a very involved solution many objects are involved. The latter will need several rules and is investigated when adaptation processes are considered. Here we consider firstly the adaptation of single objects. This is of particular interest for complex and object-oriented case representations.

A simple kind of adaptation rule is to change values of variables for those of attributes. Such a rule can change more than one value.

Examples:

- (a) Rule: The colour of white cars can be changed to red, blue or green. This rule contains only constants.

Table 9.1 A simple adaptation rule

Preconditions:
Query_#bathrooms > #Retrieved_bathrooms
Query_#bedrooms < #Retrieved_bedrooms
Conclusions obtained from applying actions:
#bathrooms = Query_#bathrooms
#bedrooms = #Retrieved_bedrooms - 1
price = Retrieved_price + 2000

(b) Add one more bathroom to a house, as outlined in Table 9.1. This rule contains two variables, #bathrooms and #bedrooms.

This action is not only concerned with the attributes occurring in the query: In addition, the attribute “price” is included, but the attribute “size of living room”, for instance, is not. Both attributes are parts of the state description, but only the first one is relevant here, i.e., its value is directly changed by the action. Such additional changes are side effects of the action.

The preconditions of some adaptation rules specify the application conditions. This could mean that they lead to a valid solution. This, however, overlooks the situation where two adaptations together provide a solution, but not any one of the two alone. We discuss this later with adaptation processes.

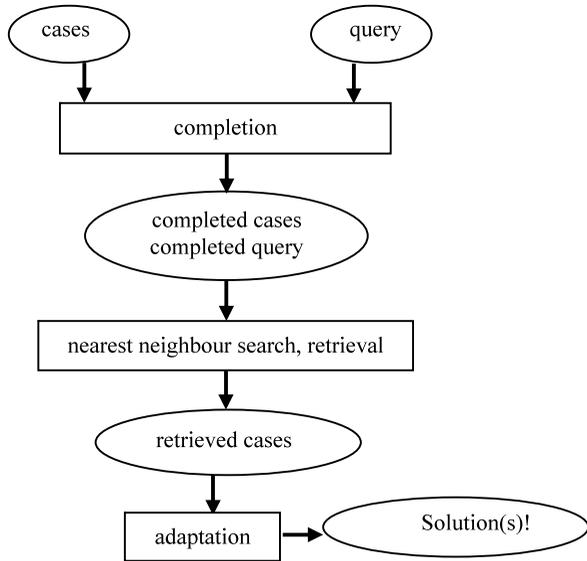
Another problem occurs when the solutions are available products. There is a difference between the situation where the set of products is finite, for instance, products in stock, or infinite. The latter occurs when the products are designed with an arbitrary degree of freedom. Then the preconditions are of a more generic character, and include instances for the variables.

9.3.4 Integrating Completion and Adaptation Rules

A specific kind of action is the creation of new objects for relational slots. This is necessary for extending the object structure of a case. For the creation of a new object, the name of the class of the object must be specified. The name of the class can be stated by specifying the name directly or by selecting a variable which is instantiated by an a-kind-of condition in the precondition of the same rule.

If the relational slot for which the object should be created is still empty then the new object is created (with empty slots) and directly linked to the slot. If the relational slot already contains an object, then this object must be of the same class or it must be a superclass of the object that should be created. If the latter is the case, the existing object is replaced by the more specific (subclass) object which is to be created, but the filled slots of the old object are directly copied into the same slots of the new object.

As an example suppose we have a robot. If we want it to see, we need to create a new object for camera equipment. An integrated view is shown in Fig. 9.3.

Fig. 9.3 Integrated view

The completion of the cases is done once while the query completion is done each time the query shows up.

9.4 Adaptation Types

Again, we start with adaptation of single objects. Typical situations occur in e-commerce where single objects provide a solution. In e-commerce a customer has a demand: A product with certain properties is desired but cannot be found explicitly in the set of listed products, which is supposed to be a subset of the product base.

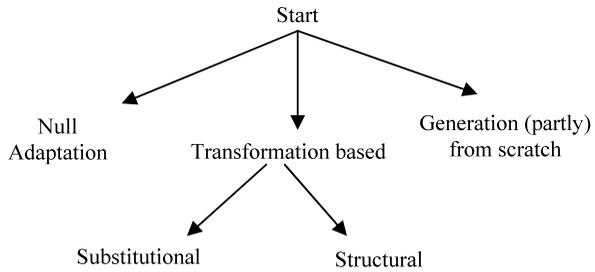
The product base P is only partially represented explicitly. Such a product representation is quite common if the set of products is very large. The commentary to the set explicitly shown explains how these products can be changed.

As mentioned, not all solutions are adaptable, and we will distinguish between adaptable and non-adaptable objects. As examples for products we have:

- Unchangeable products such as IC's, records, books, and so on.
- Products with few changeable features such as vacations, apartments, cars, and so on.
- Products with many changeable features such as computers, networks, insurance, and so on.

Automatic adaptation makes sense for configurable solutions only, and not for products in general. It will be a difficult task for solutions with many possibilities for modification and for solutions with high interaction among the parts.

Fig. 9.4 Basic adaptation types



Suppose now that the available products are split into two sets:

- The products which are directly listed in some catalogue *C* of the product base.
- The products which can be obtained by adaptation from products in the catalogue.

Often, most products have to be obtained by applying adaptation steps to products in the catalogue. Now the adaptations can be arranged in a hierarchy according to the amount of changes they provide. We will discuss several adaptation methods. An overview is given in Fig. 9.4.

The description of these types is:

- (1) Null-adaptation, for non-adaptable objects.
- (2) Transformation-based adaptations, performed by using rules.
 - a. Substitutions: They replace some part of the retrieved solution by another or by several others. This is the simplest and most common form of adaptation. Such a part could be an instance value such as the voltage of a technical device or the colour of a car.
 - b. Structural transformations: They alter the structure of the solution and reorganise the solution, for instance, by insertion and deletion of complete objects. An example is the change of travel plans when one uses a train instead of a car, for instance, when there are train stations and fixed schedules but no gas stations.
- (3) Generative adaptations: They replay the method of deriving the retrieved solution on the new problem. This is the most complex form of adaptation and we will discuss it in the Sect. 9.7 on the adaptation of more general solutions.

9.5 The Adaptation Process

Often, a single adaptation step is not sufficient. Therefore, we investigate the problem of finding a solution by making use of sequences of many available adaptations.

The problem is not as easy as when we want to change the colour of the car:

$$\text{Colour(car)} = \text{red} \rightarrow \text{Colour(car)} = \text{blue.}$$

For the colour, one adaptation step works. In general, one has to combine and iterate such steps. The problem arises that the state obtained by one adaptation step prohibits the performance of another one because some precondition is now violated.

Example I have the visitors A , B , C , and D in the morning starting at 10 a.m. Each visitor needs one hour. The constraints are that A has to be the first and D needs to be finished by 2 p.m. The solution to this problem is that the visitors come in alphabetical order. This constitutes the retrieved case.

Consider now that suddenly it becomes known that we have another visitor X who has also no time after 2 p.m. This is the new query. Consider the two plans, where X is added at the beginning or at the end; they would violate the constraints. How do we adapt them?

If we try to put X between A and B , this would require additional steps to move up C and D , which also would cause a violation. A solution can be obtained by further adaptations that lead, for instance, to the sequence A , D , X , B , C . The example shows that one has to consider the planning of adaptation processes.

9.5.1 Adaptation Sequences

Because the rules are logical rules the adaptation steps are treated as logical inferences. The relation between similarity and rules is discussed in Chap. 13, Advanced Similarity Topics. The inferences correspond to a concatenation of operator applications.

Definition 9.1 An operator sequence $op = op_n \circ \dots \circ op_1$ is a sequence of operators where the preconditions of each operator are not violated by results of the preceding operators. That means that in such a sequence each operator can be applied.

Such an operator sequence op transforms a product p into an *adapted product* p_{op} , i.e.,

$$p_{op} = op_n(p) \circ \dots \circ op_1(p).$$

The *set of all operator sequences* is denoted by OS .

In an operator sequence the ordering in which the operators are applied is important.

A typical and frequently occurring example for this is:

Suppose we have the two operators for replacing a part in a machine:

op_1 : remove the old part.

op_2 : insert the new part.

As mentioned, one need for adaptation results from the fact that CBR provides approximate solutions only and not all cases can be stored. As a consequence, there

is no exact match between what one wants and what one gets. Instead, the relation between the two objects is described by a similarity measure:

$$\text{sim}(\text{wanted}, \text{retrieved}).$$

If the retrieved solution is not satisfactory, one tries to use adaptation. Adaptation may still not lead to an exact match but it can move the retrieved solution closer to the wanted one:

$$\text{sim}(\text{wanted}, \text{retrieved}) < \text{sim}(\text{wanted}, \text{adapted}).$$

A serious problem is that for single adaptation steps, $\text{sim}(\text{wanted}, \text{adapted})$ may not increase, as seen in the example of first removing a part. In the example of finding an ordering for the visitors using permutations, one has first to go to worse orderings. Such problems originate in dependencies between parts of the solution. That means some adaptations may necessitate other adaptations in order to keep the solution valid. The reason is that there can be constraints on the solutions that are violated by the changes resulting from the adaptations.

Therefore, a simple hill climbing procedure for reaching the highest similarity will not work. Sometimes it may be necessary to go “downhill” for a while. For this reason the search for a suitable adaptation sequence is a complex problem.

The set of all objects obtainable by such inference steps from the case base CB is called the *completion* of CB under the rules of R .

In Chap. 13, Advanced Similarity Topics, it is pointed out that there is little connection between similarity and rules. This means that a high similarity between query and solution does not always admit an easy adaptation.

Consider the following situation. A store has products from two manufacturers, M_1 and M_2 . Then it can happen that the best product from M_1 is much closer to the demand than the best product from M_2 but the products from M_1 cannot be adapted at all while the products from M_2 allow adaptation. In this situation we call the products from M_1 more competent than the products from M_2 . The competence relation will be discussed in Sect. 9.5.2.8.

The reason is that similarity refers to the utility to the user while the ease of adaptation deals with the modification possibilities and costs.

Modification costs are:

- Assigned to individual adaptation steps, and
- Summed up over the path of an adaptation process.

To formulate this for similarity, a second similarity measure sim_{path} , besides the measure sim of the CBR system, is introduced.

Definition 9.2 The graph G_{path} is defined as follows:

- (i) Nodes are all products (or solutions).
- (ii) Two nodes p_1 and p_2 are directly connected if an adaptation step leads from p_1 to p_2 . The edge is annotated with the operator and its cost.

Table 9.2 Comparing measures

Similarity	Compares
$\text{sim}(q, p)$	Query q and solution p .
$\text{sim}_{\text{path}}(p_1, p_2)$	Similarity between products p_1 and p_2 by adaptation path length.
$\text{sim}_{\text{op}}(q, p)$	Query q and solution p_{op} adapted from q by op to p .

- (iii) For two nodes that cannot be connected directly the cost is ∞ .
 (iv) The measure $\text{sim}_{\text{path}}(p_1, p_2)$ is defined as the dual to the costs of the path from p_1 to p_2 .

The measure sim_{path} is a typical transformational measure like those used for graph similarities. As indicated, the two measures sim and sim_{path} will in general have different values. Note that for two nodes p_1 and p_2 with adaptation cost ∞ there may still be a path connecting them with finite cost.

What one finally wants is the nearest neighbour in the catalogue space, i.e., the whole product space, by using operator sequences from OS .

Definition 9.3 We denote this nearest neighbour by $\text{NN}(q, p, OS)$. This gives rise to a third similarity measure sim_{op} in the catalogue space:

$$\text{sim}_{\text{op}}(q, p) := \text{sim}(q, p_{\text{op}}).$$

Here p_{op} is obtained from p by applying the operator sequence op . The goal is now to find some product p in the case base CB , for which one has

$$\max(\text{sim}_{\text{op}}(q, p) | \text{op} \in OS, p \in CB).$$

The problem is that the nearest neighbour search does not only investigate all products from CB but also all operator sequences.

Now three similarity measures play a role. Table 9.2 gives an overview.

For finding the nearest neighbour after adaptation one has to search in the whole product space, which means one has to consider all of OS . The problem is that this set can be infinite. For example, in the catalogue it may be stated that for a certain object the size can be increased from n to $n + 1$ without imposing a limit on n . The problem requires careful planning what is considered next.

9.5.2 Adaptation Planning

The construction of the needed operator and the action sequence is in the domain of action planning but has special characteristics. The general problem results, in principle, in a search where the search space is the set of all the finite operator sequences. It is clear that a simple search is not feasible.

9.5.2.1 The Problem Situation

We repeat the problem situation:

- The set of cases that are listed explicitly, obtained without adaptation. This is called the listed case base.
- The set of all cases that can be obtained by applying adaptations. We call this the full case or the product space. This is the search space.

The search in the full product space may not even terminate because there may be infinitely many and arbitrary long sequences of operators. In addition, the search has to be done at runtime. As an example, consider a product that has size 10 that can be increased in size by 1, unlimitedly.

The best available solution is unknown; one has to be satisfied with a good approximation. This means that we are looking for the nearest or a close neighbour after the adaptation of a case.

The basic question is: When should we stop searching? This question can be answered in a relatively easy way: When the similarity sim between the demanded and the found solution is high enough, i.e., the solution has a sufficient utility.

For a realistic approach some knowledge about the operators, the product base and the adaptation rules is needed, in particular, prediction knowledge about how good the actions are.

There are two general methods for getting such knowledge:

- (a) Obtaining it from experts, documents, and so on.
- (b) Generating it by machine learning methods.

9.5.2.2 Solution Approaches

To find a solution in a complex search some knowledge for guiding the search is needed. Such knowledge is typically formulated in terms of suggestions or heuristics. Next, we investigate different kinds of heuristics for efficient adaptation. These heuristics are a major part of adaptation planning.

9.5.2.3 Adaptation Under Additional Constraints

The idea is to add some further restrictions in order to shrink the search space. This can be done in different ways.

- (a) If an operator op is defined on the domain of some attribute A then additional constraints can be introduced, such as “if $y \leq x$ then op can change x to y ”, and A can be equipped with some partial ordering \leq on A .
- (b) For some attributes A only certain values can be allowed. Because these attributes may be the ones for which values are demanded in the query, a local optimization has to take place in order to select the best available values or to approximate them sufficiently well.

- (c) The adaptation is only possible for specific products. This is the first situation where actually products and not only their attribute-values enter the scenario.

If one has statistics about how often certain adaptations for certain products worked, then this can lead to such restrictions.

9.5.2.4 Preferences and Lazy Adaptation

Preferences are a partial ordering on the rules that should restrict the choices for selecting them. This can be done in different ways:

- (a) Preferences can be given in terms of the affected attributes.
- (b) They can also be given in terms of operators: A partial ordering in which they are applied.

First, lazy adaptation is considered. The idea is to apply operators as late as possible, using knowledge about the result of the operator applications without executing them. The realisation is done with preference rules that state in what order the adaptations should take place.

Among the operators that could be postponed we easily find the following:

- (a) Operators *op* that have no precondition.
- (b) Operators or operator sequences *op* treating unconditionally adaptable attributes *A*. These are attributes that have the properties:

For any values $x, y \in \text{dom}(A)$ the operator sequence can yield $op(x) = y$ and no other attributes are affected by *op*.

The first condition says that the adaptation is not restricted by any constraints while the second says that there is some independence property for attributes. Both conditions will have to be weakened in order to become applicable in practice.

- (c) Operators where the application is very expensive. These operator applications may become unnecessary and one can save costs.

Typical examples of unconditional adaptations occur with operator parameters of products which can be freely chosen, but where only one specific example is represented in the product base. Other examples arise if there are changes free of charge and there are no further constraints and dependencies. Often these properties do not hold in the form stated here because at least the attribute price is affected.

In the situation of unconditional adaptation this is fairly trivial because all products remain candidates. This will change if conditions or dependencies are present.

The dependencies we consider are twofold:

- (i) The constraints for operator applications refer to other components such as the attributes, such as the one under consideration.
- (ii) The operator application has side effects that determine the values of other attributes or restrict these values.

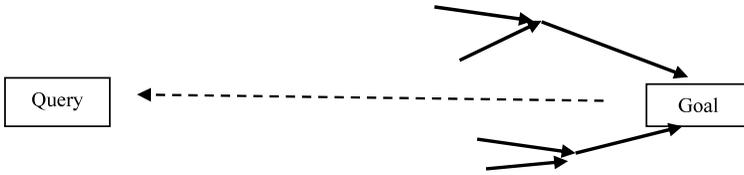


Fig. 9.5 Backward chaining

Examples If one has to plan a conference in a city and that conference has been organised in other cities too then several adaptations have to be done. For instance, determining the building has to be done early. On the other hand the booking of accommodation can be postponed if almost all participants come from the same city. It can be that there is no convenient site available for the conference in the city. Changing to a site outside the city has the side effect that transportation is needed.

In a previous diagnostic situation a certain surgery may have been done early. In the present analogue situation this may not be that urgent and can possibly be avoided. Then the adaptation can be postponed. In all such situations one has to analyse the preconditions of the adaptation rules.

An example of side effects occurred in the example in Table 9.1 for adding a bathroom in a house. In the preconditions, #bathrooms and #bedrooms are mentioned but not the price. This last attribute is, however, affected too. If there were an upper bound for the price, this bound could have been overstepped and in this case the rule could not have been executed.

Sometimes the side effect is known a priori. A problem arises when the side effects occur at runtime.

9.5.2.5 Forbidden Concatenations and Filtering

In Chap. 8, Retrieval, the MAC/FAC method was introduced. In the first phase, MAC, certain cases survive and one used only simple tests for it, like checking certain attribute-values. Therefore, this method realised a filtering step too.

This method, however, can also be used as a filter for applying adaptation steps. For instance, one could call certain adaptation rules only. This could be done for example by using constraints.

9.5.2.6 Backward Chaining

In backward chaining one starts with the goal and goes backwards using the reversed adaptation steps until one reaches the query. Figure 9.5 illustrates this.

The dashed line indicates the backward direction. This type of search can be recommended if the desired solution is known and only the adaptation process is

unknown. In addition, the search space has to be smaller than in the forward mode. This depends on the branching in the search space, where some knowledge may be available.

9.5.2.7 Using CBR

The idea is to use experience for finding suitable adaptations. This needs not only a library of cases, but also a library of cases representing episodes of case adaptation. The CBR paradigm now reads as “similar problems need similar adaptations”. This creates the need for a special similarity measure. The case base can consist of past searches of adaptations. The case base can consist of past searches of adaptations and there is a close relation to statistical methods.

9.5.2.8 Competence and Search Space Reduction

A general goal is to reduce the search space without losing (too many) solutions. As with many other methods, the analysis of competence and the introduction of a footprint has a very involved preprocessing part, leading to a comparatively simpler search for the desired adapted case. The preprocessing part tries to reduce the search space.

For the preprocessing, the following concepts are useful. We assume we have some heuristic H for adaptation. This heuristic can, for instance, be any of those adaptation strategies previously discussed.

Suppose c is a case (a solution or a product) and q is a query; OS is the set of all operator sequences. The next definition introduces the concepts for describing the competence of a CBR system.

Definition 9.4

- (a) $\text{Reachable}(OS) =$ Set of all cases that can be obtained by adaptation from the case base.
- (b) $\text{Reachable}_{\text{ad}}(p) =$ Set of all cases c that can be reached from p when adaptation is used.
- (c) $\text{Coverage}_{\text{ad}}(c) =$ Set of all problems q that could be solved from c when adaptation is used.

These concepts require a definition of the term “solve”. This can be done in different ways, depending on the application. A standard way is to require for a solution $\text{sim}(q, p) \geq \theta$ for some user-defined threshold θ .

The coverage is a measure of the competence of a case. However, this has to be determined with care. The usefulness of a case depends on the competence of competing cases too, as discussed below.

In a competence set each case must share coverage with some other cases in the set.

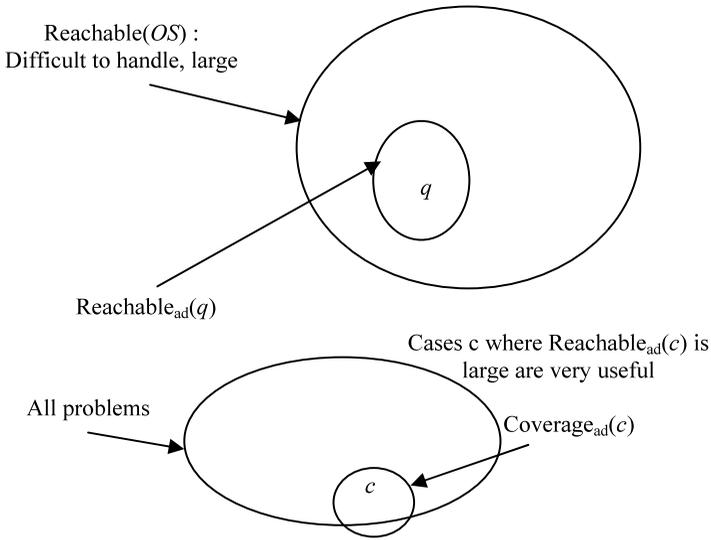


Fig. 9.6 Reachability

This leads to the concept of reachability, which is shown in Fig. 9.6.

Both sets, $Reachable_{ad}(q)$ and $Coverage_{ad}(c)$, can be infinite. Therefore, heuristics that restrict the application of adaptation rules play a major role. The usefulness of heuristics is determined by how they navigate between the two conflicting requirements:

- Efficient handling of adaptation.
- Finding a good solution.

The size of coverage of c is a measure of its competence in the sense that problems can be solved by adaptation using this case c . However, this considers just the coverage of this case. If there are other cases then they may cover the coverage(c) of c as well, which makes c redundant. On the other hand, a small coverage can still be useful if it fills a gap not covered by other cases. This regularly happens in call centre applications where one has thousands of such individual cases, though in call centres adaptation is mostly done by humans.

This motivates our considering the relative coverage of a case compared with other cases. Such relations are measured by the sets $SharedCoverage$. They provide a way of linking related cases together.

Definition 9.5

$$RelativeCoverage_{ad}(c) = \sum_{d \in Coverage_{ad}(c)} \frac{1}{|Reachable_{ad}(d)|}$$

As a consequence, if a case c is covered by n other cases then each of the n cases will receive a contribution of $1/n$ from c to their relative coverage measures.

The fact that two cases c_1 and c_2 are related with respect to their coverage is expressed by the predicate SharedCoverage:

Definition 9.6

$$\text{SharedCoverage}(c_1, c_2) \Leftrightarrow (\text{Coverage}_{\text{ad}}(c_1) \cup \text{Reachability}_{\text{ad}}(c_1)) \cap (\text{Coverage}_{\text{ad}}(c_2) \cup \text{Reachability}_{\text{ad}}(c_2)) \neq \emptyset.$$

Cases with shared competence compete with each other to find a solution. They can be grouped together into competence groups.

Definition 9.7 A subset $CG \subseteq CB$ is a competence group \Leftrightarrow

- (i) For each case $c \in CG$ there is another case $d \in CG$ such that the predicate SharedCoverage(c, d) is true.
- (ii) CG is maximal with respect to condition (i).

Each competence group makes a unique contribution to the competence of the case base. Each competence group is structured as a partially ordered set (CG, \geq) according to the size $|\text{Coverage}_{\text{ad}}(c)|$.

The next step is to reduce the case base to a subset with the same competence. This will be a footprint set.

Definition 9.8 A footprint set of a case base is a subset of the case base that covers all of the cases in the case base.

Each competence group must be represented in the footprint set but must not contain necessarily all of its members. For example, cases whose coverage set is completely subsumed by the coverage set of another case can be omitted.

The preprocessing consists of the construction of the footprint set. First compute all the related concepts. This generates a footprint set for a competence group. The total footprint set is the union of all footprint sets for the competence groups.

The purpose of the footprint sets is to simplify retrieval after adaptation. The retrieval now has two steps. Suppose footprint set FP and a query q are given.

Step 1: Search in the footprint set FP and find a case c .

Step 2: Search in $\text{Coverage}_{\text{ad}}(c) \cup \text{Reachability}_{\text{ad}}(c)$.

Both steps are relatively inexpensive compared with the search in the whole case base. The footprint algorithm computes the footprint set:

Input: Competence group (CG, \geq) ,

Variables:

FP footprint set candidate

CH Boolean predicate for changes

```

G competence group candidate
FP := {}; CH := true, G := CG
While CH := true
  For All c ∈ CG Do
    CH := false
    If FP cannot solve c Then
      CH := true
      FP := FP ∪ {c}
      G := G \ {c}
    End If
  End For
End

```

The footprint set is typically much smaller than the case base, and therefore the retrieval process is less expensive. There is, however, a problem with the deletion of cases because it may reduce the competence of the whole system. The problematic cases are the pivotal ones.

Definition 9.9 A case is pivotal if it is reachable by no other case but itself:

$$\text{pivotal}(c) \leftrightarrow \text{reachable}(c) = \{c\}.$$

Pivotal cases are general outliers and too isolated to be solved by any other case. Target problems falling within the region of a pivot can be solved only by that pivot. Therefore, deleting pivotal cases reduces competence.

Advantages of footprints:

- Most of the work is done in the preprocessing.
- Promotes fast retrieval.

Disadvantages:

- Preprocessing can be too expensive.
- Domain knowledge cannot be incorporated.

9.5.3 Learning Heuristics

In the general situation one is confronted with the same problems as a robot positioned in a labyrinth that has to find some place. The possible paths for the robot correspond to the possible applications of rules. Although a general solution of such problems is impossible, in many cases, heuristics can be very helpful. Heuristics can result from background knowledge, as just indicated. If such knowledge is not sufficiently available, methods of machine learning can be employed.

The problem can be regarded as a learning problem where one learns from experiences, i.e., from examples. This is related to making use of CBR, but learning by other methods can be used too. This will be discussed in Chap. 10, Evaluation, Revision, and Learning.

9.5.4 *Adaptation in More Complex Situations*

Sometimes the adaptation processes are insufficient because the solutions of the problems are too involved. We investigate two kinds of situations. The first one makes use of several cases and the second one makes use not only of the solution but of the whole solution process too. These approaches are related to each other. Both decompose the problem into parts in order to find partial solutions that are integrated later on. The first method deals with simpler situations where the partial solutions can be provided by cases. The second one is more involved.

9.6 Adaptation Using Several Cases

There are different reasons for using several cases for problem solving. A major reason is that there is no case that fits completely into the problem description. First we present a very simple one and then we discuss two more involved ones.

9.6.1 *Simple Numerical Adaptations*

They can work if the problems and the solutions are of numerical character. One distinguishes between interpolation and extrapolation. These are mathematical computations; from the CBR point of view they are not very involved. For that reason they are just briefly sketched by an example.

Problem: Suppose you want to build a five-story house and get an estimate of the cost. Assume the two nearest neighbours have four and six stories, with

Case₁: $\text{cost}(\text{four-story house}) = 200,000$

Case₂: $\text{cost}(\text{six-story house}) = 280,000$

The actual problem is: $\text{cost}(\text{five story house}) = ?$

Because of the metric distance in the problem for the two cases a simple interpolation is plausible and yields $\text{cost}(\text{five story house}) = 240,000$.

If the problem is $\text{cost}(\text{eight story house}) = ?$ a simple extrapolation would give $\text{cost}(\text{eight story house}) = 360,000$.

In general, interpolation and extrapolation are more complex and fairly advanced topics in numerical analysis.

9.6.1.1 Decomposition

After numerical adaptations, the next kind considered is when a problem can be decomposed into subproblems that can be treated independently or where only a small number of dependencies occurs. Another kind is when there are different cases that contribute competence to solving a single problem.

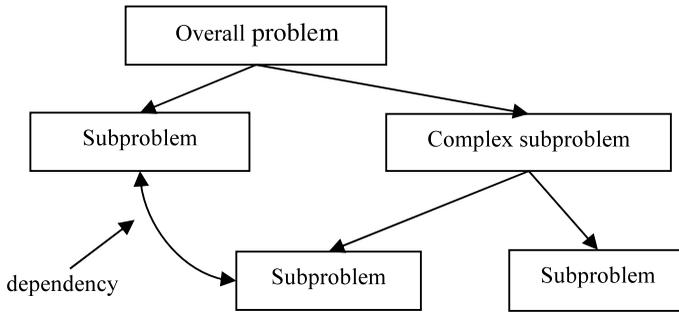


Fig. 9.7 Subproblem decomposition

This type of problem occurs in the configuration of objects with several parts that demand different kinds of experience. This requires the decomposition of the problem into subproblems as in Fig. 9.7.

The idea is to solve the subproblems independently and integrate them at the end. For each problem, different cases can be used. The solutions can then be adapted separately for a new situation with separate adaptation rules in the same way as previously discussed. However, there can be dependencies that have to be taken care of. They occur when the adaptation rules affect attribute-values that are shared by different subproblems.

The general method has the following steps:

- (1) Retrieve a solution candidate for the complete configuration problem.
- (2) Determine a still-unsolved subproblem.
- (3) Retrieve a suitable sub-solution using cases.
- (4) Try to integrate the sub-solution into the overall solution.
- (5) Repeat (2)–(4), until a solution for the overall problem is obtained.

This is illustrated in Fig. 9.8. The overall query has two parts. Some part of the query using the circle is solved using Case 1, but leaves two squares as subproblems to solve. Case 1 has two triangles as the solution. The part that remains to be solved from the query is split up. This is partially solved by Case 2, and the rest is solved by Case 3. Finally the partial solutions are integrated for providing the final solution. What is not shown in Fig. 9.8 is that the used cases may undergo adaptation.

The use of several cases occurs typically in manufacturing complex objects with many parts. For instance, different cases may be used for electrical and mechanical parts. One can easily instantiate Fig. 9.8 in this way.

9.6.1.2 Different Competencies

Here competence is understood in the ordinary sense of everyday language: Some agents may know more on a subject than others. Situations with varying competence occur also in classification and in diagnosis. The different participating cases are

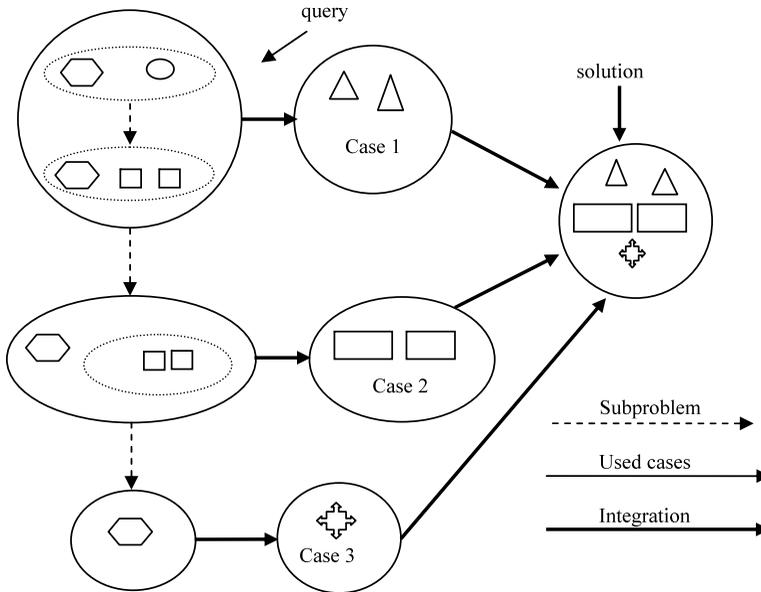


Fig. 9.8 Use of several cases

regarded as expert opinions. The problem of merging such opinions is standard in social science. Formally, the different contributions can be regarded as votes.

Suppose the problem space is divided into k subspaces where different competencies are needed, and the case base is divided accordingly. These subbases are thought of as k experts with different competencies, although they may be able to give some answer to every query. If a query is presented, there will be k -nearest neighbours. Which one should we take? Distributed case bases are discussed in Chap. 12, Advanced CBR Elements.

We also remind the reader of the voting technique for choosing the solution introduced in Chap. 6, Basic Similarity Topics. Here the voting can make use of the desired competence of the cases. In addition, two cases can be used for different parts of the solution. For instance, in a medical situation one case can have a higher competence in the diagnostic part and the other case may be more reliable for the therapy.

A key problem is determining the subbases SB_i of cases when a set of k problem spaces is given. The idea is to build these subbases incrementally as growing sets. This can be done in the following way, where a set PS_i , $1 \leq i \leq k$, of problem spaces is given.

Sketch of the algorithm:

```

Var Integer PointsPSi(c), Const  $\theta, \tau, \rho$ 
Initially, each  $SB_i$  is empty.
FOR all cases  $c$  DO

```

```

FOR all  $PS_i$  DO
  PointsPSi( $c$ ) = 0
  FOR all  $q \in PS_i$  compute  $\text{sim}(q, c)$ 
    IF  $\text{sim}(q, c) \geq \rho$  THEN test solution of  $c$ 
    IF test result is GOOD THEN
      PointsPSi( $c$ ) = PointsPSi( $c$ ) + 1
  FOR all  $c$  and  $i$ : IF PointsPSi ( $c$ )  $\geq \tau$  THEN  $c \rightarrow PS_i$ 

```

There are two concepts that have to be made precise by the user, those of the threshold ρ and the predicate GOOD. These depend on the application and the intention of the user. Observe that a case can be in several subbases if it is competent in the corresponding problem spaces. We remark also that the algorithm can be refined in different ways.

9.7 Adaptations Using the Solution Process

More general situations considered are those where solution processes are involved. This can be interpreted in two ways:

- (a) The solution contains also the way in which it was obtained.
- (b) The solution itself is a process.

We treat both ways simultaneously because for both we have to deal with processes. We distinguish between two approaches.

Transformational approach: The solution process is not considered anymore; it is contained in a black box. In this situation rule-based adaptation can be performed.

Derivational approach: In this approach one is interested in the original solution process because it contains useful information for modifying it. Often this is, for instance, necessary for guaranteeing the validity of the solution in the actual situation. Here, rules only are generally not sufficient; at least they have to be organised specifically. An illustration is in Fig. 9.9.

Situations where the derivational approach is used can occur in any kind of decision making where a complex reasoning process is involved. This is closely related to the adaptation of processes.

The solution processes are the objects under investigation. They can be quite complex and the adaptations have to be made at several places in the process. The guiding principle is to record the reasons (mainly preconditions) of such decisions and to try to apply them to the new situation.

Typical candidates of the derivational approaches are partially ordered plans, and in particular, decision plans. Each decision is governed by some decision rule that decides about the applicability of the decision.

If a new problem is presented and a case is retrieved, we first check whether adaptation is needed and then the adaptation starts. It begins with a replay of the old plan in the new situation.

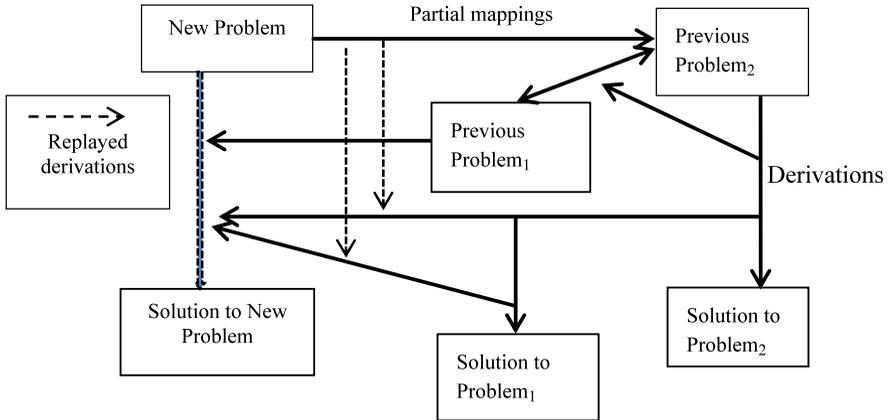


Fig. 9.9 Replay of solution processes

Replay means that the path of decisions is reconstructed relative to the new conditions.

For this, two steps have to be taken:

- (1) Recording the solution process, by defining a trace T of the reasoning process for obtaining the solution.
- (2) Instantiating the trace T in the new situation.

In the trace the types of decisions are recorded. It is useful that the case does not only record the actual decisions but also alternatives because they may be used in a new, slightly different situation.

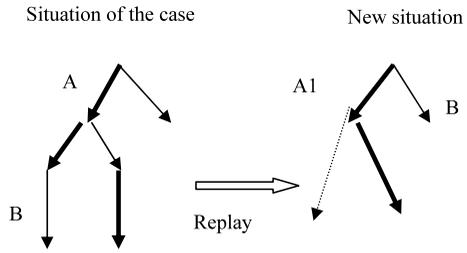
The outcome of the decisions and the processes leading to them constitute the cases.

The details of this are many and varied and we will discuss this in a very general way only. The following steps have to be taken:

- One first looks at the sequence of decisions of the case.
- At each step one tries to use the same decision rule by instantiating the variables in the new situation. This leads in principle to the same kind of decision but it is adapted according to the new instance, and the decisions are also different.
- It can also happen that the preconditions of the decision rules are no longer satisfied in the new situation.
- This presents a problem in its own right. It may be solved by using other cases or in a generative way.
- As a consequence, one does not obtain directly a complete solution but rather a guide for solving the problem. This guide provides the solution path and can reduce a problem to a routine one by avoiding finding an innovative solution. Such guidelines occur frequently in business.

Next, we see an abstract example trace with alternatives and a replay in Fig. 9.10.

Fig. 9.10 Replay



A valid step is denoted by \longrightarrow . In the new situation, *A* is possible but is instantiated differently from *A1*. \longrightarrow denotes alternatives and $\cdots\cdots\longrightarrow$ indicates invalidity.

In the replay, *B* is invalid because a precondition in the decision rule is violated. *B* is replaced by *B''*. *B''* can be obtained in different ways, for instance, by adapting *B* or by using one of the alternatives. One can also use (several) cases.

This kind of trace can be extended in various ways. Examples are:

- As reasons and justifications for the decisions.
- As reasons for not using some alternative.
- As dependencies between decisions.

The latter play a role in planning. If some alternative was not used, the reason for this may not apply in a new situation.

The major differences between the transformational and the derivational approach are indicated in Table 9.3.

In a more specific view it becomes clear that there are deeper and more problematic issues involved. A major point is dependencies. Often, they are not visible but they are a major reason for failures of replay. Typical examples are again (as in the situation of several cases) manufacturing problems of complex objects with many parts.

Table 9.3 Transformational versus derivational

Transformational	Derivational
No knowledge about the solution paths in the cases needed	Knowledge about the solution paths in the cases required
No problem-solving knowledge needed	Problem solving knowledge required
Adaptation knowledge about the result needed	Adaptation knowledge about the individual solution steps required
Correctness not guaranteed	Higher degree of correctness achieved
Oriented towards adaptation of structured objects	Extended to the usage of processes
More commonly used	Less often used, recommended for complex plans.

9.8 Quality Issues

The solutions of a problem have two aspects:

- (1) The intended functionality. This is a minimal requirement.
- (2) The quality of the solution.

The second aspect adds something to the adaptation problem. It extends towards an optimization problem: Finding a product that matches all demands as well as possible. It is not wrong to say that quality is a part of utility and is modelled in the same way. It has a relational aspect and a functional one. Quality will also be discussed in Chap. 12, Advanced CBR Elements.

Definition 9.10 Quality is a real-valued function Q from problems and solutions:

$$Q : P \times S \rightarrow \mathfrak{R}.$$

The value set \mathfrak{R} of Q is a totally ordered set of quality values (typically the set of real numbers). Quality is a more fine-grained measure than just correct/incorrect.

To approach this from the viewpoint of adaptation, we split the solution search into two parts:

- (1) Finding a solution satisfying the demands using adaptation.
- (2) Improving the quality of the solution by modifying the adaptation process.

For this purpose we regard the adaptation process as a solution plan in its own right and refer to the previous remarks on the derivational approach. In particular, one has to look at alternatives in such a derivation. The major problem is again that one cannot expect that each adaptation step improves quality and one needs an adaptation process.

9.9 Knowledge in the Adaptation Container

In this chapter the rules for the adaptation container have been split into completion and adaptation rules. This could be interpreted as a split of the container into two sub-containers: the completion and the solution adaptation container.

The knowledge in the completion container is manifold:

- How to compute values of virtual attributes.
- How to correct queries: The first step is to discover inconsistencies in the query, which requires a consistency checker.
- How to discover redundancies and implausibility.
- How to discover missing arguments.

The completion of the queries contains, if properly done, a huge amount of knowledge. One kind of knowledge is concerned with the possible demands of the

user who poses the query. The other kind concerns the interest of the institution that gives the solution. Because there are different solutions possible for the query, answers that are of highest interest to these institutions should be preferred. For example, a sales company is interested in selling specific products and a medical doctor wants the most effective therapy.

The knowledge in the completion container is interested not only in user aspects but also in technical aspects.

In the solution adaptation container one finds, first all, possible adaptation rules and their preconditions. That means it completes the case base such that all solutions and products are reachable. In addition, knowledge about adaptation processes can be stored here. This knowledge is concerned with the solution processes and has to contain much domain knowledge to be useful.

Finally, knowledge about footprints can be located also in the adaptation container. This knowledge can reduce the search space. Therefore, it is a kind of retrieval knowledge.

9.10 When Should Adaptation Be Considered?

Query adaptation and completion is in many situations in e-commerce and diagnostics not only useful but also necessary. Often, only a few products are listed explicitly and therefore adaptation is necessary. It is important to guide the user through the large set of possibilities during the search. In e-commerce adaptation is a decisive economic factor.

Adaptation plays also a role in call centre applications. The operator who receives the call almost never finds a convincing solution in a case base. Therefore, often a part of the adaptation is done by the human operator. This support here is very helpful. In both situations customers provide frequently incomplete and conflicting information, making completion very useful.

9.11 Tools

Typically, every CBR tool allows the formulation of adaptation rules and some also of completion rules. The computer support for creating adaptation plans is, however, rather limited.

9.12 Chapter Summary

In this chapter the reuse phase of the CBR cycle is investigated. It allows us to reach not only the solutions that are directly stored in the case base but also those that are available by adaptation. Adaptation is described by rules. Two kinds of rules

have been considered, completion rules and solution adaptation rules. The first type describes query completion and correction while the second describes adaptation of solutions. Iterating adaptation steps has led to adaptation processes. The search for adequate adaptation processes was observed as one of high complexity. For a realistic approach, several heuristic methods have been considered. Some heuristics are summarized as lazy evaluations. For the analysis of the search space for adaptation we used the competence concept that has led us to the footprint method for reducing the search space.

The adaptation process was extended in different ways. They included the use of several cases and derivational adaptations where the whole solution process is considered. Here the distinction between the transformational and the derivational approach was investigated. The latter is of particular interest for very complex problems.

9.13 Background Information

Not all CBR systems and tools use adaptation. Two early systems illustrate this: ARCHIE2 (Domeshek and Kolodner 1993) performs no adaptation while CHEF (Hammond 1989) does. A comprehensive overview of earlier work on adaptation can be found in Hanney et al. (1995); see also Muñoz-Avila and Cox (2008).

There are several approaches that use CBR for getting adaptation rules. The system DIAL (Disaster response with Introspective Adaptation Learning) described in Leake et al. (1995) contains an adaptation component for CBR. A special approach for learning of adaptation rules using CBR can also be found in Li et al. (2009).

Interpolation and extrapolation was extended to symbolic domains involving adaptation in Chatterjee and Campbell (1993). A general software architecture for adaptation in CBR from the methodological point of view was introduced in Plaza and Arcos (2000). In Fuchs et al. (1999) a unified approach to case adaptation is addressed by studying the issue of plan adaptation.

Footprints were introduced by Smyth and McKenna (1999) and we follow their presentation.

The distinction between transformational and derivational adaptation was introduced in Carbonell (1983). Replay originated in Veloso (1994). It was extended in Muñoz-Avila and Hüllen (1995).

9.14 Exercises

Exercise 1 Suppose you have a shop that sells books. You have three files for prices: incoming prices from publishers, internal price list, Web catalogue prices. Formulate adaptation rules for when publishers change prices.

Exercise 2 Suppose you sell cars and the car manufacturer describes the car with all technical details. Complete the description by introducing attributes that are useful for the customer and define corresponding completion rules.

Exercise 3 Describe a trip where an accident causes the closure of a road and it is hard to find an adapted plan that allows you to be on time for your return flight.

Exercise 4 Write a detailed plan for organising a conference in a large city. Use derivational adaptation to transform it into the same conference in a small village.

Exercise 5 Consider a shop that sells houses. Suppose there are several rules for adaptation where each rule application increases the price by some amount. Add a program that accumulates the prices for applying several rules. Illustrate this by some examples.

References

- Carbonell JG (1983) Learning by analogy: formulating and generalizing plans from past experience. In: Michalski R, Carbonell JG, Mitchell T (eds) *Machine learning: an artificial intelligence approach*. Springer, Heidelberg, pp 137–159
- Chatterjee N, Campbell JA (1993) Adaptation through interpolation for time-critical case based reasoning. In: Wess S, Althoff K-D, Richter MM (eds) *Topics in case-based reasoning. EWCBR-93: first European workshop, Kaiserslautern, Germany, 1–5 November 1993. Lecture notes in artificial intelligence, vol 837*. Springer, Berlin, p 221
- Domeshek E, Kolodner JL (1993) Using the points of large cases. *Artif Intell Eng Des Anal Manuf* 7(2):87–96
- Fuchs B, Lieber J, Mille A, Napoli A (1999) Towards a unified theory of adaptation in case-based reasoning. In: Althoff K-D, Bergmann R, Branting LK (eds) *ICCB-99: case-based reasoning research and development. Third international conference on case-based reasoning, Seon Monastery, Germany, July 1999. Lecture notes in computer science (lecture notes in artificial intelligence), vol 1650*. Springer, Berlin, p 104
- Hammond K (1989) *Case-based planning: viewing planning as a memory task*. Academic Press, Boston
- Hanney K, Keane MT, Smyth B, Cunningham P (1995) What kind of adaptation do CBR systems need? A review of current practice. In: Aha DW, Ram A (eds) *AAAI fall symposium. Technical report FS-95-02*. AAAI Press, Menlo Park, p 41
- Leake DB, Kinley A, Wilson DC (1995) Learning to improve case adaptation by introspective reasoning and CBR. In: Veloso MM, Aamodt A (eds) *ICCB-95: case-based reasoning research and development. First international conference on case-based reasoning, Sesimbra, Portugal, October 1995. Lecture notes in computer science (lecture notes in artificial intelligence), vol 1010*. Springer, Berlin, p 229
- Li H, Li X, Hu D et al (2009) Adaptation rule learning for case-based reasoning. *Concurr Comput, Pract Exp* 21:673–689
- Muñoz-Avila H, Cox MT (2008) Case-based plan adaptation: an analysis and review. *IEEE Intell Syst* 23(4):75–81
- Muñoz-Avila H, Hüllen J (1995) Retrieving cases in structured domains by using goal dependencies. In: Veloso MM, Aamodt A (eds) *ICCB-95: case-based reasoning research and development. First international conference on case-based reasoning, Sesimbra, Portugal, October 1995. Lecture notes in computer science (lecture notes in artificial intelligence), vol 1010*. Springer, Berlin, p 241

- Plaza E, Arcos JL (2000) Towards a software architecture for case-based reasoning systems. In: Ras ZW, Ohsuga S (eds) ISMIS 2000: foundations of intelligent systems. 12th international symposium. Lecture notes in computer science, vol 1932. Springer, Berlin, p 601
- Smyth B, McKenna E (1999) Footprint-based retrieval. In: Althoff K-D, Bergmann R, Branting LK (eds) ICCBR-99: case-based reasoning research and development. Third international conference on case-based reasoning, Seeon Monastery, Germany, July 1999. Lecture notes in computer science (lecture notes in artificial intelligence), vol 1650. Springer, Berlin, p 343
- Veloso MM (1994) Planning and learning by analogical reasoning. Lecture notes in computer science, vol 886. Springer, Berlin

Chapter 10

Evaluation, Revision, and Learning

10.1 About This Chapter

The first part of this chapter is basic and is of interest to readers who want to build or maintain a CBR system. The second part provides background knowledge. It deals with revising the methods if something is definitely wrong and with improving CBR systems by machine learning if the results are weak. The objects of improvement are in principle all processes of the CBR cycle and all knowledge containers. This chapter includes specific and general changes. In Chap. 11, Development and Maintenance, this will be regarded from a practical point of view. Here we provide the principles and foundations. The reader is expected to be familiar with Parts I and II.

10.2 General Aspects

10.2.1 *The Purpose*

This chapter is the basis for the next chapter, Development and Maintenance, Chap. 11. Both are aimed at creating and improving CBR systems. Here we provide the principles and algorithmic foundations. In Chap. 11 these are used in a practical and organisational way. In Fig. 10.1 this is illustrated.

The foundations provide algorithms, programs and tools that are used in the practical phase by a team. This team is organised and does not have to know the foundations in detail. The connection between the parts is provided by a knowledge engineer. This agent advises the various team members on which techniques to use in which way. This is explained in Chap. 11.

10.2.2 *Principal Aspects*

Revision is a step in the CBR process model. It is concerned with the fact that certain solutions to problems, although recommended by the CBR system, may fail. CBR

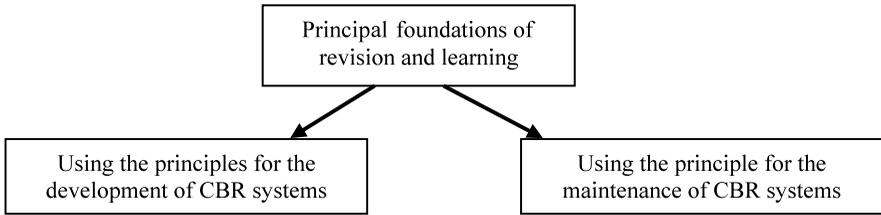
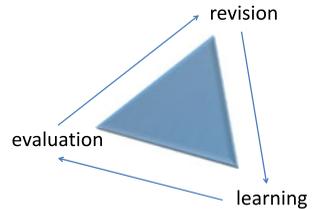


Fig. 10.1 Principal aspects and practical use

Fig. 10.2 Evaluation, revision, and learning triangle



systems like other software systems may have defects or weaknesses that need to be discovered, improved or repaired. The general improvement of a system is more of a global character.

The terms in the chapter title are in logical order: The evaluation finds out how good the results are, the revision corrects concrete failures and the learning improves general weaknesses. There is no sharp boundary between revision and learning.

Revision and learning depend on experiences too, for instance, from identifying errors when using the system or from observing a bad performance. In the discussion on the extended use of CBR it was remarked that not all experiences for using CBR are contained in the case base. They can be compiled into other containers too. In this chapter this approach is considered systematically and in detail. The evaluation may concern just one problem (as suggested in the process cycle) or, systematically, many problems.

In Chap. 11, Development and Maintenance, this will be treated from a practical point of view. In this chapter we will present the necessary background.

The terms evaluation, revision, and learning are also connected in a small cycle. See Fig. 10.2.

Firstly, the triangle requires the detection of errors and weaknesses of the solutions. This detection can be done with tests in a model or checks in reality. It can also be the result of user complaints. For example, this is done by applying tests. After testing, the other two activities follow in their logical order.

Evaluation and testing discover the weaknesses. The difference between revise and learning is essentially:

- Revise is a local step in the sense that it improves a single case with respect to some query. The usual policy is to change the case as little as possible.
- Learning is more global; it improves many cases and processes by treating more global aspects.

These points are related to each other in the following way:

- Discovery of an unsatisfactory solution gives rise to a change in the solution.
- Discovery of weaknesses give rise to activities for improving the system, which means activities that employ learning.

The starting point is always about how weaknesses can be discovered. Discovery can occur in different ways, for instance:

- Complaints about failures.
- The observation that one does not know how to get certain knowledge.
- Weaknesses compared with competitors.
- Systematic search.
- Random discovery.

Some of these are of organisational nature and will be discussed in Chap. 11, Development and Maintenance. Often however, discovery is quite an involved learning procedure. This is because many data have to be analysed. Such data contain hidden information about weaknesses that learning has to make explicit.

Some investigations on the solution have been done in the reuse step. In addition, the revise step tests the solutions from an outside institution. If the solution is not satisfactory it will be repaired. Usually, this cannot be done by adaptation because the weaknesses cannot be foreseen.

A systematic evaluation can result in the discovery not only of weak individual solutions but of weaknesses in the whole CBR system. This requires more overall tests.

The retain step adds a case resulting from the revise step to the case base if this seems to be appropriate. This can be regarded as a learning step. This learning tries to improve the case base. However, in the process model there is very little learning involved and the whole process step may even have disadvantages because the case base is always increasing. In addition, this does not cover adding cases from the outside. In addition, forgetting and modifying cases is not yet included. In this view we regard the retain step as a placeholder for more involved learning activities.

10.3 Evaluation

All evaluations provide some kind of feedback. The major types are shown in Fig. 10.3.

Now this will be discussed in more detail.

- (a) The evaluation by a teacher necessitates a (human) expert. Such experts may not be available and may also be susceptible to errors. This can be improved if the expert is not a human but a certified recorded document.
- (b) Evaluate in real world:
 - The system can notify us of relevant changes in the environment that are related to earlier problem solutions, as a result of an integrated real-time sensor system or some other feedback from reality.

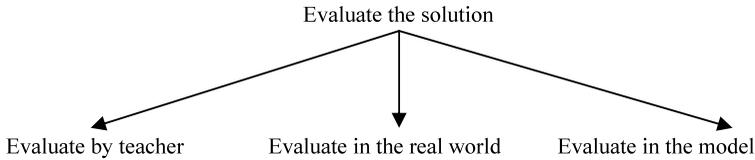


Fig. 10.3 Evaluation possibilities

- If there is also a function to judge whether the solution was appropriate for the problem or not, then the system can—to some degree—guide itself for the following integration tasks.

(c) Evaluate in model:

There are now two ways on two different levels of abstraction.

- The first is to find inconsistencies in symbolic models. If an inconsistency is found in the retrieved solutions then it has to be removed. The problem is that this can be done in different ways if the inconsistency has several participants. As an example we consider a situation where we have five cars owned by three persons and everybody owns at least two cars. This is an inconsistency that can be resolved in different ways.
- One can try to avoid reality tests by investigating solutions with numerical simulation. This requires evidence categories based on evaluation functions that determine whether a solution works or not.
- Statistical evaluation. This requires the availability of many examples where the validity of the results can be checked.

The testing needs two concepts:

- Evaluation criteria.
- Testing methods.

The testing methods split the given samples into training samples and test samples. Most common are cross-evaluations and “Leave One Out”. The latter is one of the quasi-standard methods. Leave-One-Out cross-validation (sometimes LOOCV) is done by taking one sample and using others as test cases each time. As a result, each case has $(n - 1)$ comparisons.

10.4 Revision

As mentioned, the revise step is concerned with a single problem and its solution. Because revising is based on real-world observations, the question that comes up is, where does the real-world experience come from? For this reason it is helpful to record the source of the observation. For instance, a case should be of the form:

Case = ((problem), (solution, event occurred, other)).

In other words, the experience tells us what happened when the experience was applied in the past.

This may be satisfactory or not. It tells us what a revision should do, namely, improve the used experience.

If the outcome of the application of the solution is recorded in the case (for instance, as “event occurred”) then sometimes the revise can be performed at runtime. In this case revise takes place locally and can be regarded as a part of the reuse step.

A systematic treatment of these questions is provided in Sect. 12.5.4 on provenance.

Formally, revision is an action of the form

revise: Solutions \rightarrow Solutions.

The difference with the adaptation action is that adaptation is part of the system and is a priori defined. On the other hand, the revise action comes from outside and is used if one runs out of adaptation. It is not presented by rules. The application of revise leads to a new case:

(problem, revise(solution)).

Because revision deals with one case only, it seems to be a fairly simple task. In fact, in many situations this is true. But quite often there are many problems involved:

- (1) One does not know the reason for the failure of the solution.
- (2) The reason for the failure may apply to many cases, which calls for a general repair procedure.
- (3) There may be a valid solution in the case base which the similarity measure did not find.
- (4) Even if the reason for the failure is known the repair itself may be a very difficult and complex problem requiring much expert knowledge.

For such reasons the knowledge for performing revisions is not contained in any of the containers of the system and comes from outside sources.

We will present two general ways to realise revise.

(a) Self-repair:

- The system can generate repaired solutions by itself. This may be based on, for example, general domain knowledge or simulation.
- The system is notified that a solution (or a product) has changed. This should result in a change of the solution in the corresponding case. For automating this one needs a running system in the change management of the company.

Example Suppose a case recommends the use of a database DB that is stored in a file F . If the database in F is replaced by a database DB' then the case base can be notified automatically to replace DB by DB' .

(b) User repair:

- The user must find the repaired solution. The user may be supported by automated devices. Such devices can, for instance, be search engines, which may then be integrated into adapted structures.

(c) Validity measure:

- There is a way to “measure” validity of the knowledge structure. This includes mainly finding contradictions. Similar events are changing prices or other features in an e-commerce shop. Validity is more concerned with learning and will be discussed below.

After revision, the question is what to do with the old case. It is not applicable anymore. But the revise step does not need to exclude a case totally. It can be deactivated temporarily because it may be useful for other purposes such as statistical evaluations. It may also be useful for maintenance purposes if the environment has changed.

All these topics are investigated systematically in Chap. 11, Development and Maintenance.

10.5 Learning

While revision usually takes place on demand for specific reasons, learning is of a more general character. The indicators of the need for learning are manifold, for example:

- Expensive mistakes.
- Missed opportunities, and results below expectations.
- New technologies, aging technologies, and productivity changes.
- New businesses, and improved competitiveness.

If such indicators come up then evaluations have to be performed. This is part of maintenance. In addition, such learning takes also place when a new system is developed and one does not know precisely how to do it.

The purpose of all learning is to improve something. Learning is of particular interest for situations that are not well understood or in which uncertainty is not easily encompassed. As mentioned, learning is closely related to system development and maintenance and will be used by it.

In promoting CBR, a major argument is that one can start with a relatively simple system that can do some job. The paid price is that the job usually is not done very well and the system has to be improved. This is the underlying motivation to apply learning, in particular, when the system is already in use.

One distinguishes between eager learning and lazy learning. The results of eager learning are compiled into the system for later use. This takes place, for instance, when similarity measures and weights are learned. Lazy learning means

that results are to some degree learned only when they are used. The results are interpreted at runtime. This is in principle the case for CBR in general. In particular, cases are understood and interpreted at runtime only. Nevertheless, parts of a CBR system can be improved by eager learning and compiled into the system.

In CBR, a first natural question is: Can we improve the structure and performance of the knowledge containers? We will investigate this question step by step and look at the individual containers. The learning methods in this chapter are of two kinds:

- (a) Learning without feedback: Nobody tells us what is good or bad.
- (b) Learning with feedback: Someone judges our activities.

The two major goals of learning are:

- (1) Correctness of the solution (i.e., favour a solution that is as good as possible).
- (2) Efficiency.

There can be many other goals depending on the application, such as user-friendliness, explanation service and so on.

If someone is talking about weaknesses and improvements, he needs to know “what is good” or what is “better”. This often cannot be formulated precisely, and even learning procedures cannot achieve this fully. Often, such a learning task is far too complex to be performed in reasonable time. This means one has to live with some kind of inexactness. As a consequence, the notion of being successful is weakened in the way that total correctness is given up.

There are two ways to doing this. In the first, one can give a threshold ε for an error in the learned result R . This means that such errors are tolerated. The second way is to not always respect this tolerance, but just most of the time, i.e., with a probability of at least $1 - \delta$.

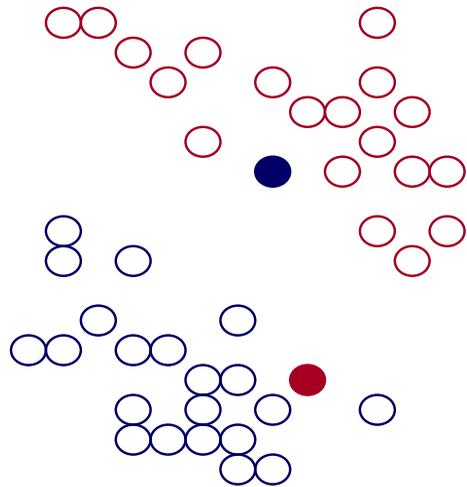
To be precise, an error function is introduced and the requirement is that for the learned result R one have

$$\text{Prob}(\text{error}(R) \leq \varepsilon) \geq 1 - \delta.$$

There are user-defined thresholds ε (error parameter) and δ (confidence parameter). This type of learning has the name PAC learning, meaning “Probably Almost Correct”. It formulates a general principle that can be varied depending on the application. It is of particular interest for CBR because one cannot determine the parameters of a CBR system exactly; CBR is mainly approximation-oriented.

This can also describe how close a subset X of cases to the whole case set CB is. We assume that CB is contained in a real space.

Definition 10.1 X is an (ε, δ) -approximation of CB if for each case c except for some set Y with $|Y| < \varepsilon$ there is some b in X such that $d(c, b) < 1 - \delta$.

Fig. 10.4 Overfitting

10.5.1 Overfitting and Underfitting

If one wants to learn something like a concept, a weight or a measure, a natural desire is to learn it from the observations as accurately as possible. There is, however, a danger involved with this that has two kinds of appearances.

The first kind is called overfitting, which means that one tries to be too exact. This is problematic when the data are not fully exact, for instance, because of noise or errors. A learning method that is “very exact” may pay too much attention to the noise and the errors. As a result, one gets a very complex concept. Overfitting is especially dangerous because it can easily lead to predictions that are far beyond the range of the training data.

Example We want classify the data into two classes as shown in Fig. 10.4. Here we see two marked errors. If we proceed with them as valid data, this would lead to two wrong class definitions.

Even if the data were correct the class definition would be somewhat complicated so that one could decide on how to deal with a slightly inexact definition that is easier to handle.

One reason for overfitting besides noise is the existence of irrelevant attributes in the training data; that is, the data may be showing misleading regularity stemming from the unimportant attributes.

The other danger is underfitting. That means there is something missing that is needed for understanding. A method that is not sufficiently complex can fail to fully detect important data in a complicated dataset, and this can again lead to an unwanted error. Such a model performs poorly on new examples as it is too simplistic to distinguish between them (i.e., the model has not picked up the important patterns from the training examples).

In a statistical situation, underfitting produces excessive bias in the outputs, whereas overfitting produces excessive variance.

10.6 Learning to Fill and Modify Knowledge Containers

Although the ultimate goal is to improve the performance of the whole system, it is necessary to investigate the individual containers too because the changes leading to an improvement take place.

10.6.1 *The Vocabulary Container*

Filling this container means finding terms that are useful or even necessary for the task in question. The automated extension of the vocabulary is almost impossible because of the huge number of possibilities not restricted by any constraints; it is an open world. Today, the acquisition of new vocabulary is usually still a creative process that can only be carried out appropriately with intensive help of domain experts.

Nevertheless, there are ways to improving a given vocabulary in general. We present three methods.

- (a) The first one is removing irrelevant attributes. In machine learning this is known as feature selection. Redundant attributes may lead to overfitting as well as to unnecessary complex similarity measures and retrieval. Therefore, removing unnecessary attributes is an important step. The main problem is their detection, for what one has different ways. When one uses linearly weighted measures, the relevance of an attribute is determined by its weight. Therefore, one can delete attributes with weights below some user-determined threshold τ . Weight learning is discussed in Sect. 10.6.3.
- (b) The second one is dependencies between attributes. This is a difficult task. There is some tool support; see the later Sect. 10.8 on tools.
- (c) The third is finding virtual attributes. They have been introduced in Chap. 5, Case Representations. The relation to similarity measures was discussed in Chap. 6, Basic Similarity Topics, where it was shown that they can simplify the definition of similarity measures significantly. As just mentioned, this cannot be fully automated. The detection of dependencies and the introduction of virtual attributes are closely related.

10.6.2 *The Case Base Container*

In the view of the general learning goals, there are two conflicting demands on a case base: On the one hand, it should be well informed for providing good solutions, and on the other hand, it should be small for efficient retrieval. We concentrate on classification although the methods work more generally. For this, some useful properties of cases have to be defined.

Definition 10.2

- (i) A case-based system $S_1 = (CB_1, sim_1)$ is *better informed* than the system $S_2 = (CB_2, sim_2)$ if S_1 classifies more *problems correctly* than S_2 .
- (ii) A case base CB of a case-based system (CB, sim) is called *minimal* if there is no sub-case base CB' of CB s.t. (CB', sim) that classifies at least as many cases correctly as (CB, sim) does.

The goal can now be formulated for a fixed similarity measure as:
Find a case base CB that is:

- (1) As informative as the whole set of given cases;
- (2) Minimal for this property.

Of course, a minimal set may not be unique. There may be, for instance, two cases with the same class close together and distant from all other cases; then, either of the two can be deleted.

In this context it is useful to consider the set of given cases as a cluster and case bases as sub-clusters. The goal can then be seen as finding a minimal cluster with the conditions formulated on the predictive power in Chap. 6, Basic Similarity Topics. This is the target of the first learning methods considered here. They are of simple character and the principle is not to fill all cases in the case base. Although they can deal with any similarity measure, in the examples we take the Euclidean distance.

The situation considered is one in which a set of cases $\{C_1, C_2, \dots, C_n\}$ is given. There are three basic algorithms, IB1, IB2, and IB3, for inserting the cases into the case. The term IBL is an abbreviation of "Instance-Based Learning".

IB1, the simplest method, is the most primitive form of learning, which does not really deserve the name: It takes all cases into the case base. The disadvantage is that the case base grows fast and will be far from being minimal. It may store unnecessary cases. One is interested in not storing all cases.

A question that remains is: How many training examples are needed for IB1 in order to get with *high probability* an *almost correct* description of the intended concept?

A theoretical result is the Covering Theorem:

"If the cases are represented in the real plane and contained in a bounded set B , and C is a concept, then there is a number n_0 such that if a training set T has at least n_0 many elements then the set determined by IB1 is an (ε, δ) -approximation of C ".

The number n_0 can be computed explicitly. This means that the wanted concept C is obtained with a high probability and it is approximately correct. As a result, if we have an unlimited number of available cases for classification, we can find the result in a satisfactory way.

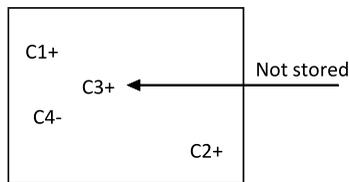
The method IB2 refines this by taking cases only if the actual case base performs a misclassification. The algorithm is again simple.

Initialize: $CB = \{\}$

FOR $i = 1$ to n DO

 Take $C_i = (p, c)$ where $p =$ problem and $c =$ class

Fig. 10.5 Weakness of IB2



```

Choose from  $CB$   $C' = (p', c')$  s.t.  $NN(C', C_i)$ .
IF  $c \neq c'$  THEN  $CB := CB \cup \{C_i\}$ 
 $CB := CB \cup \{C_i\}$ 
END
    
```

The problem is now that the results of IB2 depend on the presentation sequence, although if a training case is correctly classified in the training phase, and is therefore not taken into the case base, it may very well happen that this case is misclassified in the final case base. In Fig. 10.5, four cases are presented; the minus and plus sign indicate the class. Case C3 is not stored, which leads to an error because the nearest neighbour for case C1 is now case C4, which is in the wrong class.

The result is that IB2 stores much fewer cases. In experiments, it was shown that IB2 has a classification performance almost as good as that of IB1.

The third method, IB3, deals with the quality of individual cases. In contrast to IB2 it not only takes cases if the actual case base performs a misclassification, but also it removes *bad* cases. The algorithm IB3, which extends IB2, follows.

```

Initialize:  $CB_{acc} = \{\}$ 
FOR  $i = 1$  to  $n$  DO
    IF exists  $c' \in CB_{acc}$ 
        THEN choose  $c' = (p', s') \in CB_{acc}$  s.t.  $NN(c_i, c')$ 
        ELSE choose randomly a case  $c' = (p', s')$ 
        IF  $C_i = (p_i, s_i)$  and  $c_i \neq c'$  THEN  $CB_{acc} := CB_{acc} \cup \{c_i\}$ 
    FOR ALL  $c^* = (p^*, s^*)$  from  $CB$  with  $\text{sim}(p, p^*) \geq \text{sim}(p, p')$  DO
        IF  $\text{Bad}(c^*)$  THEN  $CB_{acc} := CB_{acc} \setminus \{c^*\}$ 
    
```

We illustrate removing “bad” cases in Fig. 10.6.

In the IB3 method, two predicates occur that are undefined at this point:

- $\text{Acceptable}(c)$ with the intention that case c should enter the case base
- $\text{Bad}(c)$ with the intention that c is significantly bad and should never enter the case base.

Both are quality predicates. There are different possibilities for defining them; we apply the view of rough sets from bad to acceptable on a scale in Fig. 10.7.

The following definition is related to the concepts of precision and recall from information retrieval, considered in Chap. 23, Relations and Comparisons with Other Techniques; here they are formulated for cases. The concepts in the definition rely on statistical data and can be estimated fairly well if enough cases (for evaluation) are available.

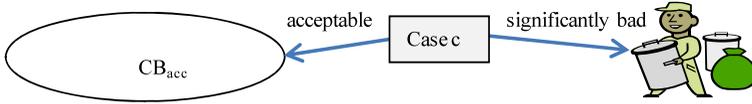


Fig. 10.6 Acceptable cases

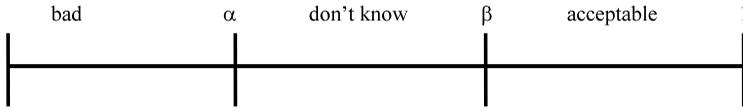


Fig. 10.7 Acceptance scale

Definition 10.3 We take a case c .

- (i) The call probability is $Call(c) = Prob(c \text{ is a nearest neighbour for an arbitrary problem } p)$
- (ii) The precision of c is $Prec(c) = Prob(c \text{ gives a correct answer} \mid c \text{ is a nearest neighbour})$
- (iii) The usefulness of c is $Usef(c) = Prec(c) \cdot Call(c)$

These definitions refer to the actual case base CB (and not to the set of all cases). Therefore, these predicates are always updated in the loop of the algorithm. The goal is to learn a base CB_{acc} consisting of acceptable cases only.

A possible way to define this is the analogue to the precision as the percentage of correctly classified objects:

$$Precision(c) = \frac{|correct \text{ classifications using } c|}{|classifications \text{ using } c|}$$

- If one takes the numbers α, β from Fig. 10.6, this leads to the definitions:
 $Acceptable(c) :\Leftrightarrow Precision(c) \geq \beta$
- $Bad(c) :\Leftrightarrow Precision(c) \leq \alpha$

The parameters α and β control the IB3 algorithm.

Discussion of the methods: The size of the case base generated by IB2 is smaller than the one obtained by IB1. IB2, however, is more sensitive to noise than is IB1. Here, noise means that the attribute-values of the class descriptions are replaced with a randomly selected value from the attribute's domain. Now, noisy instances are, naturally, almost always misclassified. Therefore, IB2 saves noisy and misclassified training instances. This leads again to generating wrong classification decisions.

IB3 is a step toward improving a weakness in retain phase of the CBR cycle: IB3 not only adds cases, but can forget some of them. Now it should be remarked that the quality of the retrieved solution depends heavily on the initially offered cases. If they are biased then the case selection is of little help.

The learning performance of all three algorithms is sensitive to the number of irrelevant attributes used to describe cases. If irrelevant attributes are ignored, this

effectively reduces the dimensionality of the search space. As previously mentioned, in linearly weighted measures, irrelevant attributes are those with low weights.

Advantages:

- The algorithms are easy to implement (with the possible exceptions of the predicates acceptable and bad).
- IB2 reduces the case base significantly with a tolerable error.
- IB3 can handle noise.
- For classification it is quite efficient.
- Learning can be influenced by knowledge.

Disadvantages:

- The methods do not consider adaptation.
- IB2 results depend on the ordering of the input cases.
- Small concepts may have a higher inaccuracy when learned.
- IB2 is sensitive to noise.

10.6.2.1 Forgetting Cases

The IB2 and IB3 algorithms do not take all cases and attack forgetting cases in this way. However, they are not directly concerned with learning by explicitly forgetting. This will be our next concern.

The goal is to forget cases without losing competence. This is of particular interest when adaptation is involved. We recall three concepts from Chap. 9, Adaptation:

- $\text{Reachable}_{\text{ad}}(q) = \text{Set of all cases } c \text{ that can be reached from } q \text{ when adaptation is used.}$
- $\text{Coverage}_{\text{ad}}(c) = \text{Set of all problems } q \text{ that can be solved from } c \text{ when adaptation is used.}$
- $\text{Pivotal}(c) \Leftrightarrow \text{Reachability}(c) = \{c\}$

Pivotal means that if c is the query, there is no other case that can solve the problem of c . Hence, forgetting c would reduce the competence. Unnecessary cases can always be forgotten; this means precisely:

Definition 10.4

$\text{Unnecessary}(c) \Leftrightarrow \exists c' \in \text{Reachability}(c) \text{ such that } \text{Coverage}(c) \subset \text{Coverage}(c').$

Here c can be deleted without reducing the competence. These are two extreme situations that can be weakened in different ways, depending on the importance associated with the cases.

Advantages:

- Easy to implement.

Disadvantages:

- Problematic, and extensions needed for non-classification or fuzzy classification.
- Quality depends on the cases offered. This is problematic if there are many or potentially infinitely many cases.

10.6.3 *The Similarity Container*

Learning the similarity measure has two targets:

- (a) Learning local measures.
- (b) Learning global measures.

For both, we distinguish between learning based on different criteria.

(1) Feedback:

- Without feedback: statistical analysis of the data.
- With feedback, in general by some kind of teacher.

(2) Granularity:

- How precise are the weights learned, for instance, real weights?
- Assumptions on the case presentation.
- Generality: For all cases, class- or case-specific.

The formal aspects of similarity measures, and, in particular, for evaluating its success, are mainly developed for classification tasks. The extension to more general tasks has mostly not been done in the same precise way.

10.6.3.1 Learning Without Feedback

We start with an elementary method without feedback and consider a representation with n numerical attributes. Then cases are points in n -dimensional space. The basic idea is to cluster the spaces in order to get insights into which dimensions have an influence on the clusters. Dimension reduction always leads to a simplified similarity measure. We show a simple example in two dimensions for classification into two classes in Fig. 10.8.

In fact, A_1 is sufficient for deciding about class membership and the second attribute can be deleted. The principle behind this method is that one detects clusters where only one dimension is involved. This can be extended for more general situations using clustering algorithms.

10.6.3.2 Learning with Feedback

The only feedback one can expect from a user is of a qualitative form, i.e., in terms of similarity relations. Users can judge them but there is no teacher who knows the

Fig. 10.8 No feedback

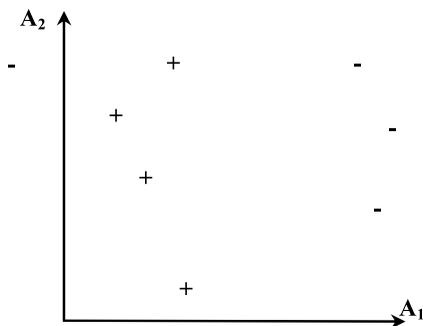
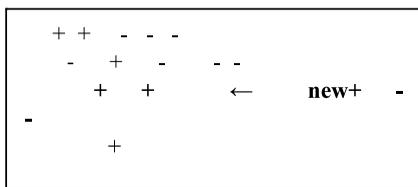


Fig. 10.9 Improving the measure



precise form of the values of a similarity measure. From this kind of learning one can get information about

- similarity relations,
- weights,
- local similarities.

10.6.3.3 Learning of Similarity Relations

Learning of similarity relations improves the qualitative aspect of similarity measures. The simplest way is to directly correct specific errors in the nearest neighbour search. Below, this is extended to the more qualitative partial ordering, \geq .

The improvement of correctness in one step is depicted for classification into two classes in Fig. 10.9. The new case is classified wrongly and the improved measure should have the effect of moving in the direction of a class with a correct solution.

The improved similarity measure “moves” the cases in the sense that cases have now a new similarity relation. This is for weighted measures the same as introducing elliptic queries; see Chap. 6, Basic Similarity Topics. However, this may have a temporary effect only because the move may not be good with respect to later cases. This is the same situation as in the IB2 algorithm: The result depends on the presentation ordering.

Now some more details are given. If a query q is provided, the CBR system arranges the solutions in a partial ordering with respect to their similarity to q where k -nn returns the first k :

$$\text{sim}_{\text{ord}} = s_1 \geq s_2 \geq \dots \geq s_k \geq \dots \geq \dots .$$

The user or expert now returns a feedback in the form of a possibly different sequence, say:

$$feedb_{ord} = s_3 \geq s_2 \geq \dots \geq s_5 \geq \dots \geq \dots .$$

The advantage of this kind of feedback is that often humans can provide it even if they do not have a definition of the ordering they have in mind. We observe that this considers a qualitative improvement of the similarity measure only, not a numerical one. However, if one is simply interested in the first nearest neighbours (which is usually the case), this is completely satisfactory.

However, one may ask the question: Why should I be interested in such an ordering if I only want to retrieve the nearest neighbour? The answer is twofold. Firstly, the nearest neighbour property is a relation too; the relation to the second nearest neighbour can be very small. Secondly, the approach can save evaluation time. If one poses the query, one gets not only the nearest neighbour, but also the other neighbours almost for free. Considering them will help improve the similarity measure to a larger degree than looking just at the nearest neighbour.

There are different ways to define an error function. They all compare the computed ordering sim_{ord} and the feedback ordering $feedb_{ord}$. For this, we make use of the edit distances graph introduced in Chap. 7, Complex Similarity Topics. These distances are used for determining the error size.

A very simple measure was the permutation distance:

$$d_{perm}(sim_{ord}, feedb_{ord}).$$

More involved distances can distinguish the positions where the permutations take place. One can have the permutations at the beginning of the sequence more expensive than those at the end. Such refinements can be determined by using appropriate cost-sensitive edit distances. In order to make use of a comparison, an error function as well as a correction method have to be defined. The correction method has to minimize the error.

10.6.3.4 Learning Weights

Next, we come to global measures. For the global measure we consider weights in linear measures. The importance of finding weights has already been emphasized in Chap. 6, Basic Similarity Topics. It contains a method for directly determining weights, which, however, will not always work. Therefore, learning of weights is of interest.

A very general form of weight learning uses a form of reinforcement learning. A learning step looks like:

- (1) Perform a test with the solution: This provides the feedback.
- (2) If the outcome is positive, give a positive reward to the weights.
- (3) If the outcome is negative, give a negative reward to the weights.

The rewards are generally of the form:

$$\text{Reward} : \omega := \omega + \Delta\omega.$$

Now, we have to consider the performance. The tests measure the performance of the CBR system. This can be done in many different ways. We evaluate the nearest neighbour.

Suppose the test case is (p, s) and the nearest neighbour case is (p_0, s_0) .

The two kinds of rewards are:

(1) Reward = positive (for correct behaviour):

- Weights of attributes with the same values for p and p_0 are increased.
- Weights of attributes with different values for p and p_0 are decreased.

(2) Reward = negative (for incorrect behaviour):

- Weights of attributes with the same values for p and p_0 are decreased.
- Weights of attributes with different values for p and p_0 are increased.

The effect of these rewards is that the test case moves towards its nearest neighbour for a positive reward and moves away from it for a negative reward.

Suppose that ω_{ik} is the present weight for attribute A_i , and a_i is the value of attribute A_i in case c . Then, set as an initial step:

$$\omega_{ik} = \text{Prob}(\text{class}(c) = k | c_i).$$

This is the conditional probability for a class k of c under the condition of attribute-value a . One can estimate the probabilities by using “samples” of the case base.

A major problem is that the rewards consider one query only. For the following query, it may happen that just the opposite rewards are given and no asymptotic judgment can be made. This phenomenon is well known in machine learning.

Therefore, it seems better to consider a larger set of queries simultaneously. This set should be representative of all possible queries.

If one considers a large set of queries, it is usually impossible to generate starting vectors in such a way that good results can be expected. A way out is to generate a small set randomly. This is done in several machine learning methods, in particular, genetic algorithms, which are discussed later.

The learning of weights can be interleaved with learning the case base in the IB3 algorithm. At each step one can compute the weight of an attribute as the predictive power introduced in Chap. 6, Basic Similarity Topics:

$$\omega(A) = \text{PredPow}(A).$$

This can also be used for updating the weights if the cases are presented incrementally. A difficult problem is learning local similarities because they contain domain knowledge.

A major problem is that users make their choice on personal preferences and do not compare it with other measures. One way of learning local measures is by using genetic algorithms. We will show this in Sect. 10.7.3 on genetic algorithms.

10.6.4 The Adaptation Container

The role of rules is discussed in detail in Chap. 9, Adaptation. There are many learning algorithms for rules and in general they are not specific to CBR. A very popular and efficient way to represent rules is in the form of a decision tree (introduced in Chap. 8, Retrieval). From the perspective of adaptation rules, the annotations at the leaf node are adaptation actions. In the nodes going to the leaf nodes the preconditions are listed.

In a commercial catalogue, the adaptation actions are usually presented in a specific way. This means one is told that in this or that situation one can perform a certain adaptation. What one is often missing are general types of rules that show alternatives and can be iterated in a systematic way.

This can be done in a decision tree. One benefit is that you can immediately see alternatives. In addition, the tree can be ordered with respect to the costs of verifying the predictions. This can be a decisive advantage.

Besides this, one can use a rule-based system to replace a part of the CBR system. This is discussed in Chap. 9, on adaptation.

10.7 Applying Machine Learning Methods

Besides the learning methods discussed so far, there is an abundance of more learning methods. This section is intended for readers who have knowledge about machine learning and want to get some information on where and how it can be used for improving CBR systems. This is no attempt to introduce you into machine learning.

Because machine learning methods are often useful for improving CBR systems, they apply to all knowledge containers. Most methods are very well designed and tested and they are useful if the situation is complex and little human knowledge but enough material data is available. Machine learning methods are of a quite different type. On the other hand, CBR systems have many different aspects that could be improved. A basic question is therefore when and for what purpose a certain learning method should be selected. In the next section we sketch a number of such methods.

10.7.1 Regression Learning

This is a supervised form of learning. Regression is a technique for minimizing the error when a function f is computed; here we think of f as a classifier. The general

error function for the goal function f and the estimate f^* is defined for the k -nearest neighbours c_1, c_2, \dots, c_n as:

$$E(c_k) = \frac{1}{2} \cdot \sum_{i=1}^n f(c_i) - f^*(c_i)^2.$$

The standard situation is when f is linear. The way to minimize the error is to use the gradient descent method. Regression is of interest to CBR when the cases are coded as points in the real space. The error between the correct function f and actual function f^* can then be assumed to be function of the weights. This can be used for weight learning because the weights are the arguments of the error function.

A simple way of the weight correction is again of the form

$$\text{Correction : } \omega := \omega + \Delta\omega.$$

A more complex approach uses the first k -nearest neighbours. These cases are combined together as a set constituting a generalized case. The input and output attribute-values of it are calculated by weighting them according to their similarity to the target case. Then a linear regression function is applied to predict the output difference between every two cases. This can be used for combining the weights of the k -nearest into one weight.

10.7.2 Artificial Neural Networks

Artificial neural networks can be used in different ways for improving CBR systems. A very challenging task that applies to software engineering in general is that the network produces test cases for software testing.

We consider backpropagation nets. They have several input nodes and one or more output nodes. The network computes the input-output function. The edges are annotated with weights; for learning the intended function, the training changes the weights according to the actual error of the output. A more restricted application is to use the network for a sensitivity analysis of the attributes in a weighted measure. Here, the input nodes correspond to the attributes.

Recommendations for using neural networks cannot be given in general terms; they are very much application-dependent.

10.7.3 Genetic Algorithms

If there is no test available for testing a solution directly as good or bad; then there is no direct supervision possible. Often, this is the case when dealing with CBR systems. This happens because there is no general and formally precise quality measurement for the contents of the knowledge containers. In special situations, evaluations can be possible, and, in addition, one may have expert judgment.

Table 10.1 Mutation of local similarities

Sim	a_1	a_2	...	a_n
a_1	1	s_{12}	...	s_{1n}
a_2	s_{21}	1	...	s_{2n}
...
a_n	s_{n1}	s_{n2}	...	1

Sim'	a_1	a_2	...	a_n
a_1	1	s'_{12}	...	s'_{1n}
a_2	s'_{21}	1	...	s'_{2n}
...
a_n	s'_{n1}	s'_{n2}	...	1

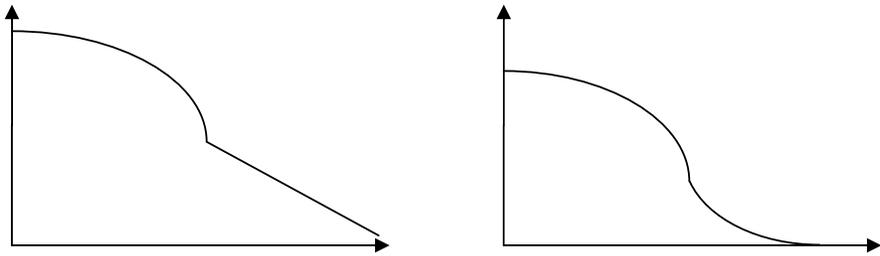


Fig. 10.10 Mutation of local similarities

In semi-supervised learning, the evaluation is replaced by a fitness test (or function). Genetic algorithms implement an iterative way to evolve (i.e., improve) a solution. They produce randomly a selection and a crossover of solution candidates, and random changes of the candidates. They select the most fit of an offspring according to a fitness test. In other words, the fitness level determines some candidates that will be eliminated and others that will survive for further processing. This can be applied to weight learning as well as to the learning of local measures. We will restrict ourselves to local measures.

The fitness function considered comes from feedback of a user who has a similarity ordering in mind, and generated measures are compared with the user feedback relation. The user always has knowledge about the local similarity measures too.

We show two examples with typical random changes that are applied for genetic algorithms. We just mention that, in the algorithm, mutation randomly picks an example and indicates what has to be changed. In CBR it is a local measure and one of its description elements. The simplest genetic operator is a mutation that changes the description element randomly, based on a mutation parameter that can change.

The first example deals with symbolic representations. Suppose attribute A has the values a_1, \dots, a_n . In Chap. 6, Basic Similarity Topics, the local similarity for A was described in a table. The original similarity is indicated on the left side in Table 10.1. The values s_{12} and s_{n2} have been randomly selected and their values have been changed due to a mutation operator, and the result is shown on the right in the table. In the same way, crossover operators are described.

Next, we consider numerical local similarities as discussed in Chap. 6, Basic Similarity Topics. An example is shown in Fig. 10.10.

On the left side we see the original measure, that is split into two parts. The second part is selected for mutation and the result is shown on the right side. This can again be done for crossover change too.

The possible advantage of genetic algorithms is that the statistical selections reduce the search time for a good measure.

10.7.4 Reinforcement Learning

This is another form of weakly supervised learning. Instead of providing a fitness value to the solution, it provides a gain or a penalty to a learning step. One can regard it as feedback from the system. Sometimes reinforcement can be used to learn an optimal weight set. Suppose we have a classification problem with two classes. We can proceed as follows.

- (1) An initial weight set is used and the output accuracy of the results is calculated.
- (2) Correct answers give rewards and false answers impose penalties.
- (3) The rewards are expressed in terms of weight changes: Weights of the attributes that caused the false positives or false negatives are decreased.

This loop continues until it reaches the minimum number of false positives and false negatives.

When the best weight set is selected, the design of the CBR system is complete and the system can enter the training phase.

10.7.5 Clustering Methods

This is an unsupervised form of learning: Given a set of objects and a similarity measure between them, the method splits the set into subsets such that objects in the same subset are similar to each other and dissimilar from objects in different subsets. The measure taken should reflect properties that distinguish objects in such a way that dissimilar objects are treated differently by the CBR system. In Sect. 10.6.3.1 we showed that clustering could help the determination of weights.

Given a dataset of customers, one wishes to build a case base to classify customers with a given socioeconomic background in order to offer them appropriate products. A clustering algorithm can help create categories for classifying each customer.

One of the consequences is that one can use special similarity measures for comparing demands and products that reflect customer preferences. However, the measure used for clustering may be simply a first guess.

Clustering is useful for learning redundant attributes. The clustering takes place in an n -dimensional space where n is the number of attributes. The analysis of the resulting clusters can lead to a dimension reduction, as illustrated in the example of Fig. 10.10.

When using clustering methods, it is important to consider that clustering is a task that is meant to create groups in datasets that have those natural groupings. If the dataset is not likely to be organised in such a way, then it may not converge. Thus, there is a variety of clustering methods. Some clustering methods (e.g., density clustering) can be used to identify outliers—those objects that are not close to any cluster. This can be used for the identification of pivotal cases.

One limitation of clustering methods, also stemming from the need of natural groupings, is the requirement posed by many methods to define the ideal number of clusters before executing the algorithm. This is also known as cluster validity. There are many ways to validate clusters. A simple and general way to validate c clusters is to verify that c clusters produce a significant improvement over $c - 1$ and are only minimally worse than $c + 1$. The parameters to determine those values may be chosen based on the problem, namely, cluster centroid, average distance within and cross clusters.

If one first clusters the cases into groups of similar cases and then applies the IBL algorithms to the clusters, that may improve the results.

10.7.6 Bayesian Learning

Bayesian inference is introduced and discussed in Chap. 15, Uncertainty. As a learning method it deals mainly with classification. Because the inference nets are goal-directed, there is mostly a relation to retrieval.

As an example of applying Bayesian nets to CBR we consider the following situation. For some attributes, it may be that the absolute value of the similarity is not as important as whether it is simply above or below a certain threshold value (i.e., in the sense of rough sets). Determining the threshold value can be instrumental for success.

Case retrieval nets are introduced in Chap. 14, Advanced Retrieval. This is in principle an influence diagram similar to that of case retrieval nets, where the edges have been labelled with similarities and weights. If the weights are replaced by conditional probabilities, one obtains a Bayesian net. This is possible only if enough probabilities are known and it could be useful if little is known of the relevancies. An advantage is also that conditional probabilities can update dynamically.

10.8 Tools

There are many machine learning tools that can be used in the CBR context too. We refer to popular and established tool collections such as Weka (<http://www.cs.waikato.ac.nz/ml/weka>) and Matlab (<http://www.mathworks.com/products/matlab/>). Standard clustering tools are k-means, density clustering and Cobweb (<http://weka.sourceforge.net/doc.dev/weka/clusteringers/Cobweb.html>).

The CBR Shell Java V1.0 contains a genetic algorithm case tool for weight learning (<http://www.aiai.ed.ac.uk/project/cbr/cbrtools.html>).

The Reinforcement Learning Toolkit 1.0 is a Python package available for multiple platforms (<http://pypi.python.org/pypi/Reinforcement%20Learning%20Toolkit/1.0>).

Weka 3 allows the visualization of the dependencies between the attributes and the relationships between all combinations of each two (with a graph including two of them on each axis).

10.9 Chapter Summary

In this chapter methods for evaluating, revising and improving CBR have been discussed in that order. Evaluation detects the weaknesses of a system. Revise is an activity from outside for removing individual faults and related problems. Learning considers improvement of the whole system. It deals with all knowledge containers of a CBR system as well as the overall system. For improving the case base, the three classical algorithms IB1, IB2, and IB3 have been introduced.

The learning of similarities was concerned with similarity relations, weights, and local similarities. As a crucial step in similarity learning, the learning of weights was identified. This task can be approached in very different ways. We considered some of them.

At the end, other machine learning methods are briefly presented. Some of these techniques are discussed with respect to their possible application to CBR.

10.10 Background Information

The evaluation methods for CBR have no essential differences compared with such methods in general software development. Of course, they are always related to the special application. Details on evaluation methods can be found in Bang et al. (2008). Statistical evaluations are discussed in Efron and Gong (1983).

An overview of revision can be found in López de Mántaras et al. (2005). There are many connections between learning in CBR and general machine learning techniques. A comparison of CBR methods and the version space method is given in Globig and Wess (1995). The IBL algorithms have been introduced and analysed in Aha and Kibler (1989) and in Aha et al. (1991). They extend earlier work on nearest neighbour algorithms which also save and use selected instances to generate classification predictions; see Cover and Hart (1967). It is also shown that the IB1 algorithm has a convergence property for describing classes C in the n -dimensional real space in the sense of PAC learning: If one has sufficiently many instances, then nearly always (i.e., with an arbitrary high probability) one gets an approximately correct definition C . The connection between IB1 and PAC is also

studied extensively in Griffiths and Bridge (1997). They combine it with similarity considerations; see also Globig et al. (1997).

Learning adaptation rules is discussed in Lieber (2007) and in Badra and Lieber (2008).

Learning similarity relations from an ontology point of view is discussed in Vazifedoost et al. (2007).

An overview of more sophisticated methods for weight learning is in Aha (1998). A PAC learning framework in this direction can be found in Satoh and Okamoto (1994).

Feature weight learning using reinforcement learning is described in Salamó et al. (2009). Learning feature weights with neural nets is in Park and Kim (2006). For learning feature weights with different neural methods, see Paikari et al. (2010).

Genetic algorithms for supporting optimizing measures were considered in Shin and Han (1999) and Stahl (2004).

The application of Bayesian learning is discussed in Kontkanen et al. (1998). There have been many attempts to incorporate Bayesian methods into CBR systems; we just mention Aha and Chang (1996) and Aamodt and Langseth (1998) as typical examples. All uses of Bayesian methods depend on the stochastic nature of the applications.

Algorithmic learning theory considers theoretical limitations of learning results, both in absolute terms as well as in terms of complexity. Theoretical results for the learnability (i.e., possibilities and limitations) of CBR systems can be found in Globig et al. (1997).

10.11 Exercises

Exercise 1 Give examples of overfitting and underfitting when describing stocks of used cars.

Exercise 2 Define rules for the bicycle domain that automatically revise cases when producers change their inputs.

Define self-repair rules for an e-commerce shop if there are changes in a price list. What is the advantage of self-repair?

Exercise 3 Define a method that identifies redundant cases.

Exercise 4 (complex) Describe a set of products in a grocery store by attributes. Suppose there are three types of customers, those who buy famous brands, no-name customers and ecologically oriented ones.

Classify these customers using a clustering algorithm. Observe that the clusters are not disjoint. Learn weights for each cluster. On their basis, define a similarity measure for each customer.

Exercise 5 Generate complaints about a favourite solution of a product or a service and formulate a revision.

Exercise 6 Show how outliers can be identified using density clustering that give rise to overfitting.

References

- Aamodt A, Langseth H (1998) Integrating Bayesian networks into knowledge-intensive CBR. In: Aha DW, Daniels JJ (eds) *Case-based reasoning integrations: papers from the 1998 Workshop*. Technical report WS-98-15. AAAI Press, Menlo Park, p 1
- Aha DW (1998) Feature weighting for lazy learning algorithms. In: Liu H, Motoda H (eds) *Feature extraction, construction and selection: a data mining perspective*. Kluwer, Norwell, pp 13–32
- Aha DW, Chang L (1996) Cooperative Bayesian and case-based reasoning for solving multiagent planning tasks. Technical report AIC-96-005, Navy Centre for Applied Research in Artificial Intelligence, Naval Research Laboratory
- Aha DW, Kibler D (1989) Noise-tolerant instance-based learning algorithms. In: *IJCAI 1989: eleventh international joint conference on artificial intelligence*, vol 1. Morgan Kaufmann, San Mateo, p 794
- Aha DW, Kibler D, Albert M (1991) Instance-based learning algorithms. *Mach Learn* 6:37–66
- Badra F, Lieber J (2008) Representing case variations for learning general and specific adaptation rules. In: Cesta A, Fakotakis N (eds) *STAIRS 2008: fourth starting AI researchers' symposium*. *Frontiers in artificial intelligence and applications*, vol 179. IOS Press, Lansdale, p 1
- Bang Y, Kim JG, Hwang I (2008) CBR (case-based reasoning) evaluation modeling for security risk analysis in information security system. In: *SecTech'08: international conference on security technology*, Hainan, China, December 13–15 2008. IEEE, Los Alamitos, p 66
- Cover TM, Hart PE (1967) Nearest neighbour pattern classification. *IEEE Trans Inf Theory* 13:21–27
- Efron B, Gong G (1983) A leisurely look at the bootstrap, the jackknife, and cross-validation. *Am Stat* 37(1):36–48
- Globig C, Wess S (1995) Learning in case-based classification algorithms. In: Jantke KP, Lange S (eds) *Algorithmic learning for knowledge-based systems*. *Lecture notes in computer science (lecture notes in artificial intelligence)*, vol 961. Springer, Berlin, p 340
- Globig C, Jantke K, Lange S, Sakakibara Y (1997) On case-based learnability of languages. *New Gener Comput* 15(1):59–83
- Griffiths AD, Bridge DG (1997) PAC analysis of similarity learning IBL algorithm. In: Leake DB, Plaza E (eds) *ICCBR 1997: case-based reasoning research and development*. Second international conference on case-based reasoning, Providence, RI, July 1997. *Lecture notes in computer science (lecture notes in artificial intelligence)*, vol 1266. Springer, Berlin, p 445
- Kontkanen P, Myllymaeki P, Silander T, Tirri H (1998) On Bayesian case matching. In: Smyth B, Cunningham P (eds) *EWCBR-98: advances in case-based reasoning*. 4th European workshop on case-based reasoning, Dublin, Ireland, September 1998. *Lecture notes in computer science (lecture notes in artificial intelligence)*, vol 1488. Springer, Berlin, p 13
- Lieber J (2007) Application of the revision theory to adaptation in case-based reasoning: the conservative adaptation. In: Weber RO, Richter MM (eds) *ICCBR 2007: case-based reasoning research and development*. 7th international conference on case-based reasoning, Belfast, UK, August 2007. *Lecture notes in computer science (lecture notes in artificial intelligence)*, vol 4626. Springer, Berlin, p 239
- López de Mántaras R, McSherry D, Bridge DG et al (2005) Retrieval, reuse, revision, and retention in case based reasoning. *Knowl Eng Rev* 20(3):215–240

- Paikari E, Richter MM, Ruhe G (2010) A comparative study of attribute weighting techniques for software defect prediction using case-based reasoning. In: SEKE 2010: 22nd international conference on software engineering and knowledge engineering, Redwood City, San Francisco Bay, CA, 1–3 July 2010
- Park SC, Kim JW (2006) Feature-weighted CBR with neural network for symbolic features. In: Huang D-S, Li K, Irwin GW (eds) ICIC 2006: international conference on intelligent computing, Kunming, China, 16–19 August 2006. Lecture notes in computer science, vol 4113. Springer, Berlin, p 1012
- Salamó M, Escalera S, Radeva P (2009) Quality enhancement based on reinforcement learning and feature weighting for a critiquing-based recommender. In: McGinty L, Wilson DC (eds) ICCBR 2009: case-based reasoning research and development. 8th international conference on case-based reasoning, Seattle, WA, July 2009. Lecture notes in computer science (lecture notes in artificial intelligence), vol 5650. Springer, Berlin, p 298
- Satoh K, Okamoto S (1994) Toward PAC-learning of weights from qualitative distance information. In: Aha DW (ed) Case-based reasoning: papers from the AAAI workshop. Technical report WS-94-01. AAAI Press, Menlo Park, p 128
- Shin K, Han I (1999) Case-based reasoning supported by genetic algorithms for corporate bond rating. *Expert Syst Appl* 16:85–95
- Stahl A (2004) Learning of knowledge-intensive similarity measures in case-based reasoning. Dissertation, University of Kaiserslautern
- Vazifedoost AR, Oroumchian F, Rahgozar M (2007) Finding similarity relations in presence of taxonomic relations in ontology learning systems. In: CIDM 2007: IEEE symposium on computational intelligence and data mining. IEEE, Los Alamitos, p 215

Chapter 11

Development and Maintenance

11.1 About This Chapter

This chapter considers the performance of CBR systems as a whole. The view is split into the initial situation, where the systems are built the first time and the situations where systems have to be maintained. The chapter is addressed to readers and institutions that develop and maintain CBR systems systematically, as in companies. The chapter is based on Chap. 10, Evaluation, Revision, and Learning, and it is thus strongly recommended you study it carefully.

11.2 General Aspects

There are many reports describing how CBR systems have been developed. Mostly, this is considered a creative task. This is analogous to the way computer programs were developed early on.

Nevertheless, software engineering techniques can give good advice on creating programs with less effort and in less time. The purpose of this chapter is to extend such advice to CBR systems and to provide the basic concepts for considering development and maintenance in a more systematic and efficient way.

This chapter is devoted to companies or institutions that regularly create and use CBR systems. For considering this systematically, organisation is needed and will be described. The task of the organisation is to make use of the principal insights to take appropriate actions.

Development and maintenance of a project are closely related. The organisational structures as well as the responsible agents for both overlap and are partially identical. In principle, development could be viewed as maintenance on a zero basis. This is, however, a very superficial view, as we will see.

In this chapter we consider mainly attribute-based systems. Other CBR systems concerned with texts and images are treated in the respective Chaps. 17 and 18. The same applies to Conversational CBR. We remark, however, that many of the principal steps described in this chapter apply to the other representation techniques too.

We provided the theoretical and methodological basis in Chap. 10, Evaluation, Revision, and Learning. There we discussed the discovery of weaknesses of systems and their repair.

We have two perspectives: A local one on knowledge containers and a global one on the overall system. For the local view we refer heavily to the material presented so far. The overall view requires integration into the processes of the company.

Usually, a company has a repository of useful knowledge. This can include:

- Experiences with previous systems
- Case bases
- Methods for learning, testing, and evaluation
- Guidelines

The repository can be helpful for all steps in the development of a system as well as for its maintenance. In general, it is part of the overall knowledge management structure of a company. We return to this later when we consider the experience factory.

Development and maintenance are not discussed in the CBR process cycle. For maintenance, there is a step in the cycle, namely, the revise step. In our presentation, we consider this step as a placeholder for maintenance in general. In general, development and maintenance refer more to institutions outside of the process model.

11.3 Development

System development is the first step one has to perform when dealing with CBR. One can refer to several knowledge sources that are provided by the institution that does the development. We proceed by discussing the steps in the process model and the knowledge containers.

11.3.1 Problem Formulation

The development problem is intimately connected with its context. This context is not only connected with the domain, such as certain available technical devices but also with the pragmatics. This considers the different intended uses; for instance, should the device be:

- Manufactured
- Sold
- Repaired
- And something else

This was considered in the introductory example on cars in Chap. 2, Basic CBR Elements. For system development it is essential to study this, to investigate user and customer demands, and to define business goals. This includes defining tests

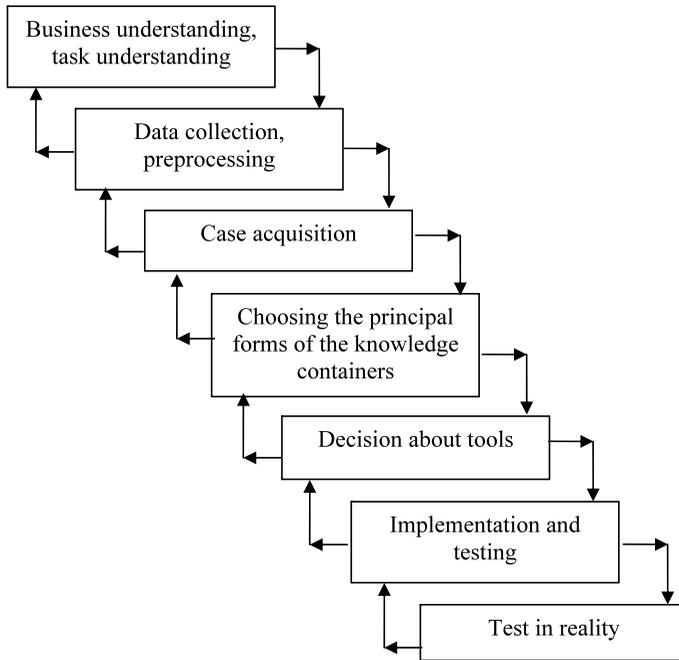


Fig. 11.1 Steps in system development

and evaluation methods. More on contexts is given in Chap. 12, Advanced CBR Elements.

System development combines general principles from software engineering and special demands resulting from the CBR process model and its knowledge containers. We will not discuss general problems of business and task understanding and software engineering principles. They are considered for the specific aspects of CBR only.

When a system is developed, two related future aspects have to be considered:

- (1) The flexibility to change the system.
- (2) The maintenance.

Both aspects have to be considered in the system development and maintenance. The techniques for both are closely related to each other. Maintenance (discussed later in this chapter) responds to changing contexts and gives for instance answers to changing demands from users. This means one wants to use the same system at a later time when the context has changed. This is of interest if several closely related developments are expected. One uses the system at a later time but for slightly different tasks. This happens frequently in commercial applications, or in medicine when different hospitals are considered.

The principal steps in the process of developing a CBR system are highlighted in Fig. 11.1.

The business and task understanding result in a requirements document. The development of a system is always a continuous process. This means sometimes one has to go backwards. This will now be discussed in more detail.

11.3.2 Finding and Getting Data, Preprocessing

The data is the raw material from which the CBR system is finally built. The collection of data is sometimes but not always problematic. The major reasons for problems with data are:

- Data may be noisy:
 - Incorrect data
 - Wrong values for the attributes
 - Incorrect classification
 - Duplicate data: They are not completely useless. They can be used for statistical purposes but not for problem solving
- Incomplete data:
 - Missing values for some attributes
 - Missing attributes
 - Missing objects
- Data not usable:
 - Terminology not understood
 - Not suitable for the intended goals

Some examples are:

- Technical or medical diagnosis: Often the past events are stored on a database. In many cases, however, the raw data are not directly usable.
- Collecting data from real-world software engineering projects is problematic. Software projects are notoriously difficult to control and corporations are often reluctant to expose their own software development record to public scrutiny. Often advices for repairing are given in a cryptic formulation.

If the data are problematic some preprocessing is required. As a first step the factors creating the problems have to be identified. There one has to distinguish the following:

- Problems that cannot be removed and one has to live with
- Problems that can (partially) be removed

Problems of the first kind arise if something is missing and cannot be collected and restored. This is, for instance, the case with medical or fault diagnosis. Often, one may not have recorded the history properly because no fault was expected. It is the responsibility of the management to take care of such things.

The techniques for dealing with problems are discussed in Chaps. 6 and 7 on similarity. Problems that can partially be handled in preprocessing are again of different types; here, methods from Machine Learning in Chap. 10 are helpful. Noisy data with relatively small errors can be handled by methods related to regression methods. For identifying outliers density, clustering is useful.

If the data are not usable, in general some kind of human intervention is needed. A proper management strategy will also enforce a unified terminology in a company that is understood by everybody responsible for the tasks. In multinational companies this can be very difficult, in particular, if several cultures are involved.

11.3.3 Case Acquisition

For case acquisition there is no fully automated method for establishing a case base. Exceptions are situations where the cases come from a database or a preexisting case base.

At first, one has to extract the cases from the preprocessed data. These data are raw material that need some refinement. The structure of the cases has to include the structures of the

- Queries
- Solutions

This requires special attention being paid to the vocabulary. In particular, the vocabulary may originally contain incorrect terms that are not understood. This leads us into the area of textual CBR and thesauri, which is considered in Chap. 17, Textual CBR.

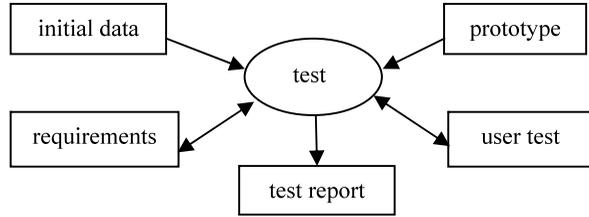
The cases available from the given data may not be sufficient. For that reason, additional cases have to be extracted from other sources or be generated. The latter activity is called case authoring. In addition, the repository may contain cases from previous and related problems. For instance, if one has a system to produce technical devices, the descriptions about the created devices are useful for fault diagnostics.

11.3.4 Prototypes and Evaluation

In the beginning, one develops a first and preliminary system, called a prototype, and evaluates it. (Please observe that the term prototype is used in the literature in different ways.) The evaluation goals should be defined in the very beginning of the development (as done in agile programming). The knowledge repository then should provide the measurement and testing tools for getting the material needed for evaluation. There are two kinds of evaluations:

- Evaluations of the overall systems.
- Evaluations of the individual knowledge containers.

Fig. 11.2 Prototype evaluation



The overall evaluation considers the prototype that is initially developed. This is shown in Fig. 11.2.

The initial datasets usually do not contain redundant data only, but they are in addition incomplete. As often is the case in software engineering, the requirements are in general preliminary and likely to change after testing. The evaluation and the testing are not CBR-specific.

One aspect of the evaluation is to find out whether the results meet the initial expectations. This may take place in the model or in the real world, and is in competition with other methods and approaches. The environment provides a teacher for this. The CBR-specific elements come up when the individual knowledge containers are investigated. The evaluation of the prototype can be regarded as a maintenance operation; it will be discussed within Sect. 11.4.

11.3.5 The Knowledge Containers

The largest part of system development comprises filling the knowledge containers and allowing the execution of the steps in the process model.

11.3.5.1 The Vocabulary Container

The choice of the vocabulary for describing cases is dominated by the business view of the users. It is important that the vocabulary be compatible with the terminology of the company or the institution.

In particular, one has to use a vocabulary understood by the potential users. This vocabulary depends not only on the domain but on the user types too. Example types are experts or laypersons. The vocabulary may not be the same as the one used in the data collection.

For the solution one has in addition to decide what elements it should contain:

- Just the pure solution.
- Additional information such as images, diagrams, sounds, explanations, experiences, and so on.
- Information about the use of the solution in reality.

This requires two activities: Selecting the attributes a case should contain and determining their values. The first results from the investigation of user demands. For the second one we can follow different strategies, for example, we can include:

- For symbolic attributes, the most frequent value (which corresponds to a default value).
- For numerical attributes, the mean value.

The given attributes are often to some degree redundant. It is the task of attribute selection to find the ones that are necessary and useful. Learning methods for this have been introduced in Chap. 10. The evaluation of the vocabulary, in particular, the understanding, can mostly be done by the users only.

11.3.5.2 The Case Base Container

The case base has two possible origins: The cases provided by the application and additional cases that are created cases. For filling the case base from existing cases in a database this process should be automated, i.e., the CBR system should have access to the database. This means, in particular, that downloading cases from a database should be automated. In addition, case authoring will take place, which creates new cases.

Initially, one takes all available cases. The given cases, for instance, the cases recorded by the user, are often to some degree redundant. For reducing the cases, see, for instance, the IBL algorithms in Chap. 10.

11.3.5.3 The Similarity Container

The similarity measure contains compiled knowledge. This is knowledge about the utility of a previous solution for a new problem. Initially, the measure will be of a simple nature. Often, the intended measure is knowledge-intensive, which provides additional difficulties.

Because the measure reflects utility, one needs first a business understanding and the formulation of goals. Often, the utility is not given as a utility function but rather as a preference relation. Therefore, one aims first for a similarity relation. The transfer from a relation to a function should go in parallel for utility and similarity.

When turning to the measures, the dependency on the used case representation has to be observed. Often, one has little influence on the representation because the structure may be given from the outside. For flat representations the task is relatively simple. For object-oriented case representations it is more difficult. The goal is to determine the similarity between two objects, i.e., one object representing the case (or a part of it) and one object representing the query (or a part of it). According to the local-global principle the object similarity can be determined recursively in a bottom-up way, starting from the attributes. This allows us to build the measure in a systematic way.

When the first prototype is evaluated, some knowledge can be shifted from the cases to the measure.

11.3.5.4 The Adaptation Container

Rules represent additional knowledge that is not in the other containers; see Chap. 9, Adaptation. We have described:

- Completion rules: They are defined during system development. They describe how missing information in the query can be provided and how misleading aspects can be corrected.
- Adaptation rules: They describe how a retrieved case can be adapted to fit the current query. Often, they come from external sources.

For defining the completion rules, one needs to check the vocabulary for the queries and select expressions that simplify understanding but are not absolutely necessary, for instance, the virtual attributes. In addition, one needs to identify default values; these are useful for case completions.

The sources of adaptation rules are often in commercial catalogues of products that indicate the allowed adaptations. The work that remains to be done is to coordinate the rules with the cases in a way that simplifies the retrieval.

The definition of additional rules is most of the time a hard job and requires background knowledge. Because this is domain-dependent one cannot give general guidelines.

11.3.6 Which Additional Methods Can a CBR System Have?

Often, a CBR system is not only supposed to present a full solution to a query. In many situations the institution that uses the system is likely to take additional advantage of the presented solutions. This can be done in many ways, for instance, with:

- Images, audio and video devices.
- Explanations and instructions on how to use the solution.
- Results from applying the solution in reality.
- Use of solution as input to a more general computer program.

For these reasons the system needs to be equipped with the corresponding support. These additions should be part of the problem solution.

11.3.7 Systematic Development of CBR Systems

This section is, in particular, addressed to persons in a company who regularly have to develop CBR systems. It is closely related to the maintenance of CBR systems, which will be discussed in the next section. Systematic development has two prerequisites:

- An organisational structure that specifies the participating agents, their roles and their relations.
- The process model for the development activities.

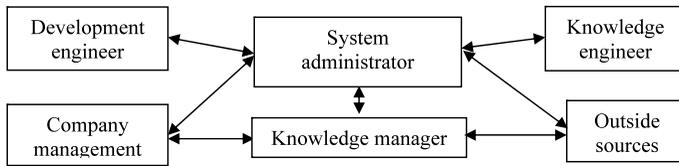


Fig. 11.3 The development team

11.3.7.1 Organisational Structure

The following agents are involved in the development of a CBR system:

- The knowledge manager is responsible for overall knowledge management in the company, which is (or should be) a part of the company management. The knowledge manager has to update and organise the knowledge repository and to make it accessible to the knowledge and development engineers.
- The CBR system administrator is responsible for system development and maintenance. This includes organisation of the flow of information. In Sect. 11.4.3.2 on maintenance we will introduce change management as a part of this structure.
- The knowledge engineer is responsible for knowledge acquisition.
- The development engineer is responsible for the technical development of the system.

The relation between these participants is shown in Fig. 11.3.

The agents are to some degree also responsible for maintenance activities. Precise relations depend on the company structure. The flow of information is discussed when change management is introduced in the next section.

11.3.7.2 Process Model for Development

For the systematic development there are three subprocesses:

1. The pure software engineering subprocess. This will not be discussed here.
2. The knowledge management subprocess. This is essential for our task and summarizes the above discussions. Here we may have again several subprocesses.
3. The development of the periphery, in particular, the user interface.

It is the task of the system administrator to instantiate the process model to a workflow for a specific purpose. A model that summarizes the process model steps, in particular those of filling the knowledge containers, is shown in Table 11.1.

Besides showing the knowledge management actions, Table 11.1 shows some additional steps:

1. User interface development.
2. System integration.
3. System test.

Table 11.1 Development process steps

Vocabulary	Represented objects	Representation format	Attribute selection	Formalization
Case acquisition	Collect cases	Select cases	Enter cases to the case base	
Similarity	Goal, utility	Principle type and definition		Formalization
Adaptation	Task	Rules		Formalization
User interface	User types	Interface types	Specification	
Integration	Integration type			
System test	Internally	By simulation	Externally	

**Fig. 11.4** Database relations

11.3.7.3 Periphery

In addition to the knowledge containers, the periphery has to be considered too. The periphery consists of all additional parts necessary for applications. Modifications to the periphery do not affect other system parts but can have an impact on user-friendliness and efficiency. Important parts of the periphery are:

- User interfaces
- Database connections
- The hardware platform

The hardware platform will not be considered here.

The position of the database in the process model is as shown in Fig. 11.4.

11.3.7.4 Database Connections

The representation of the data in the database may not be the same as the case representation. As next step, the import of cases, a transformation between the two representations has to take place. Several CBR systems support this.

11.3.7.5 User Interfaces

The user interfaces need to take into account knowledge about the users:

- What is their interest?
- What do they understand?
- In what details are they interested?

To answer these questions the user types have to be investigated. It can happen that there are two user types, for instance, beginners and experts. Instead of requiring two user interfaces designed independently, they should be designed with as much integration as possible.

A special situation occurs in Conversational CBR, Chap. 20. There, the queries are completed dynamically, which requires an interface in which windows are opened also dynamically. Those techniques are useful when further understanding or details are needed.

11.3.8 Implementation Aspects

The point of departure for the implementation is the processes in the CBR cycle. This gives rise to tasks and the task structure identifies a number of alternative methods for a task.

Method descriptions comprise the following:

- Name: The fully qualified name of the class that implements the method.
- Description: A textual description of the method.
- Context Input Precondition: A formal description of the applicability requirements of the method, including input requirements.
- Parameters: Method configuration parameters. These parameters are the variable hooks of the method implementation. For example, a retrieval method may be parameterized with the similarity function.
- Competencies: The list of tasks this method is able to solve.
- Subtasks: In decomposition methods this element provides the list of tasks that result from dividing the original task.
- Input and Output: Output data information obtained from this method execution. The information will be used to check which method can take as input the output of this one.

This is just a general view but the implementation should follow such recommendations.

11.3.9 Combining CBR with Other Techniques

So far the development of “pure” CBR systems has been discussed, i.e., systems that contained the process elements of the CBR cycle only. However, frequently a technique that is not contained directly in the CBR cycle may be needed for solving problems. There are quite a number of such techniques that a CBR system possible need to have. Some examples are:

- Database connections
- Propagation of constraints
- Systems for discovering dependencies

Table 11.2 Forms of integration

Forms	Integration basis
Toolbox	Hybrid
Common language	Both systems use the same programming and expression language
Module basis	Communication between modules
Seamless	Both techniques are invisible to the user

- A diversity of learning methods, for instance, for weights
- Other representation methods and their techniques
- Numerical and probabilistic methods
- Graphics and visualisation
- Image processing methods
- Evaluation procedures

In general, this is a standard problem in software development and can be handled in different forms of integration. The major ways of doing this for two different systems are shown in Table 11.2.

The choice of the integration type mainly depends on the role additional techniques play with respect to the CBR system. A pure toolbox level is rarely recommended because integration with other programming languages is traditionally problematic. In many cases additional programs can be connected with existing CBR tools. In fact, this is a quality criterion for CBR tools. The seamless form hides details that are not of interest to the user.

A must in this direction is to download data from a database. For testing purposes it is convenient to enter a whole set of queries simultaneously and obtain the results in the same way. Most tools do not provide this facility but it can be added easily by implementing a simple loop.

If the additional technique shares problem solving with CBR, for instance, if some problems are solved by CBR and others by decision trees, then seamless integration is recommended. In such a case a basic requirement is that the underlying data structures for representing problems and solutions be compatible.

11.3.9.1 Experience Factory

An experience factory is a logical and/or physical organisation that supports project development by analysing and synthesising all kinds of experiences and methods. It is a general repository. However, it is not aimed at experiences that constitute the case base of a CBR system. Rather, it provides a repository of experiences that are useful for system development and maintenance. The experience factory idea has been used for developing software projects for many years. It is motivated by the observation that any successful business requires a combination of technical and managerial solutions which includes a well-defined set of product needs to satisfy

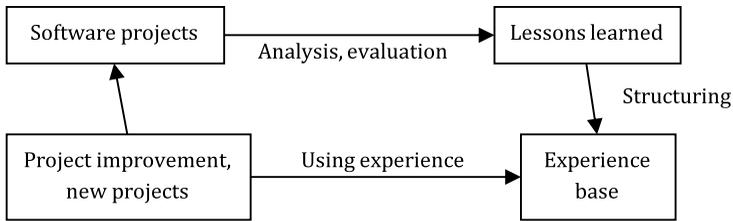


Fig. 11.5 Experience factory

the customer and assist the developer in fulfilling those needs. This includes a well-defined set of processes. The experience factory is a dynamic object that needs continuous maintenance. An experience factory can also contain ontologies. These can be concerned with domain knowledge or with development and maintenance knowledge.

An obvious question is about what the difference between an experience factory and a case base is. There are in fact several differences. First, a case is a pair (problem, solution) where the solution is usually qualified as good or bad. A general experience is something that occurred or was recorded in the past. In addition, consequences of these recordings are listed, for instance, as guidelines. In the early years of CBR that was considered as a pre-stage of a case, so to speak raw material for building cases.

Important parts of an experience factory are tools and methods to evaluate and analyse the individual experiences. Therefore, an essential part of the experience factory view is the extraction of lessons learned; this has no direct counterpart in CBR (more on this in Chap. 21). Experience factories are also closely connected to maintenance. Maintenance can benefit from the factory and also contribute to the experiences.

In this view, experience management and experience factories are less specific than CBR systems. Although the factory idea is quite general, it is useful for CBR too. An overview is depicted in Fig. 11.5.

An experience factory contains experience packages by building informal, formal or schematized models and measures of various software processes, products and other forms of knowledge via people, documents, and automated support.

Now we mention some examples of such experience packages to be used for CBR. These packages are based on the basic CBR process model and the knowledge containers.

Examples of packages:

- Available case bases
- Sources of case bases and their reliability
- Experience with complex case representations
- Used similarity measures for different purposes
- Rules for different adaptation purposes
- Experiences with texts and conversational CBR
- Experience with customers and providers

Fig. 11.6 Maintenance of technical objects

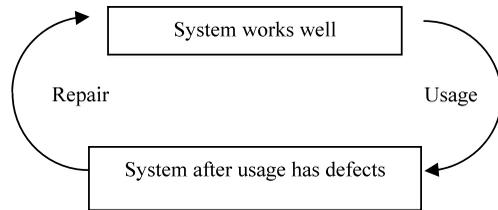
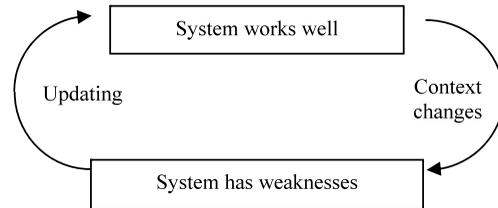


Fig. 11.7 Maintenance of CBR systems



In an experience factory these packages are systematically recorded and organised.

11.4 Maintenance

Maintenance for technical devices is standard for cars, machines, buildings, and so on. There are detailed specifications for maintenance formulated in terms of ISO 9000 norms. They contain rules and guidelines for quality management. The norms are kept general but their origins in the automobile industry cannot be overlooked. There are systematic procedures as shown in Fig. 11.6.

This shows a basic difference between the motivation for maintaining a real physical object and a software product. Physical objects need maintenance because they are used and each use lowers their quality. Software objects require maintenance because the context is changing.

For software products one distinguishes between software maintenance and knowledge maintenance. The former results from updates of the software environment and is not considered here. An elementary maintenance is the revise step in the CBR process cycle. However, maintenance is much more general.

Maintenance of knowledge-based systems is concerned with:

- Corrections: Removing bugs
- Improvement of performance
- Adaptation to changed environment and changed knowledge

Therefore, we will concentrate on changes in the environment and in the knowledge. How can knowledge change? There are areas where this does not take place. For instance, mathematical laws are time-invariant and do not change at all. But we should note that our knowledge about such topics can change significantly. In the same way, our knowledge about daily and commercial environments can rapidly change. The need for maintenance of CBR systems is represented in Fig. 11.7.

In addition to the context changes, the system may contain weaknesses or even bugs that were not discovered before the system was used. Maintenance of CBR systems is intended to improve the steps in the general process model and the overall business success. Therefore, maintenance should be a standard step in the life cycle of a CBR system. The CBR cycle will be extended in order to include this activity.

11.4.1 Changed Environment and Techniques

This section refers heavily to the revision considerations in Chap. 10.

11.4.1.1 Basic Elements

These changes are the most important but depend heavily on the application. First, we collect a number of concepts that are used for describing maintenance actions. There are different views on this.

Firstly, two categories can be observed:

- (1) Changes that require an immediate revision; we call them corrective actions.
- (2) Slowly increasing changes; we call them adaptive actions.

Corrective actions generally repair bugs or are results of specific outside changes. Adaptive changes are unavoidable because the environment and the context never remain constant. Even perfect systems need such maintenance. In principle, there are two kinds of reactions:

- Reactions on demand: Someone has a direct complaint or observation.
- Proactive reactions: One is acting before a complaint or demand arrives.

A distinction between changes is:

- Visible changes: They are the results of observations and messages.
- Invisible changes: Small changes of the context that may, however, add up.

11.4.1.2 Actions on Demand

Actions on demand result from explicit outside information or from the discovery of errors and bugs. These can be provided by the user institution or by internal institutions, for instance:

- Change in resources: Materials, tools, persons, and so on, as required for the solution are no longer available or have changed their character essentially.
- New rules in the company.
- Discovery of misbehaviour of the system.
- Customer complaints that contain error reports.
- New customers and new business.

The main purpose of actions on demand is to provide corrections that ensure that the system is working properly. In most situations an immediate reaction is necessary. However, if the corrections are small, they can be accumulated and taken care of at a later time. This applies, in particular, to small changes in the environment. All these reasons for actions are visible.

11.4.1.3 Internal Testing and Pro-action

Proactive action results usually from invisible changes and properties of the system and the context. They result in:

- Adaptations to small and long-term context changes.
- Make the system more perfect.
- Improved nonfunctional properties of the system, such as efficiency, interface, or understandability.

Because the changes are invisible, they have to be discovered. This is the task of tests and evaluations.

11.4.1.4 Maintenance Indicators

The problems with tests are, “When and what tests should be performed”? This applies to all maintenance operations too. The purpose of indicators is that their consideration can give information on needed maintenance operations. For this, additional measurements and tests are also needed. The indicators themselves can be observed. The information they contain is, however, hidden. The measurements and their evaluations make them explicit. Important maintenance indicators are:

- Insufficient and erroneous solutions. They are reported by users but can also become apparent by systematic checks.
- Case utilisation statistics can detect:
 - If some cases are retrieved very frequently, that may indicate poor case coverage and the need for additional cases.
 - If cases are not retrieved for a long period of time, they may be redundant.
- Retrieval time is too long. This may indicate poor case base organisation or the wrong choice of the similarity measure and the retrieval method.
- The CBR system is changed by methods it has, e.g., removing duplications. creates the need for checking the performance of the new system.

The operation of a CBR system can be viewed in terms of its components: internal, peripheral, and external components. The internal components are expressed in part (a) of Fig. 11.8 through its knowledge containers. The remaining components are the periphery and the problems. In part (b) of Fig. 11.8, we present the extension of this universe required for maintenance.

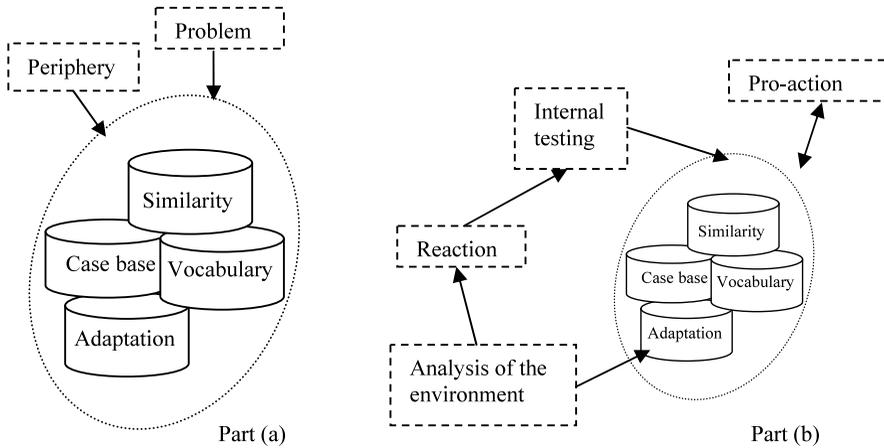


Fig. 11.8 CBR extended operations for maintenance

11.4.2 Maintenance and Knowledge Containers

Now we go down to the level of knowledge containers. The knowledge containers play an active role during problem solving, but in maintenance they are more passive objects. Here we have to pay particular attention to the relations between the containers.

11.4.2.1 Relations Between Containers

Changes in one container may affect the content of other containers because there are dependencies. Changes in the case base influence the similarity, and vice versa. Changes in the periphery have almost no direct influence on the other containers besides the fact that they may indicate the necessity of changing them. On the other hand, changes in the vocabulary have to be propagated to all other containers. We distinguish between simple updating of purely technical character, which will not be mentioned here, and more involved dependencies that require special treatment.

11.4.2.2 The Vocabulary Container

The vocabulary may need maintenance for different reasons. The main maintenance operators for attributes are:

- Change the name of an attribute. This can result from an outside demand. A possible reason is that the problem descriptions can have name conflicts.
- Change of the dom(attribute).
- Add an attribute.
- Delete an attribute.

The propagation to other containers is simple. The range change can affect the local measure. For special situations, new problems with new attributes can arise. This is the same situation as that discussed in Sect. 11.4.1 for new developments.

For the adaptation it is somewhat more involved because it can make certain adaptations unnecessary because the new term in the vocabulary no longer needs to be obtained by adaptation.

An operation of the form “add an attribute” requires a change in the similarity measure; one has to determine a weight and a local measure for the new attribute.

The attribute changes can have a major impact on the user-friendliness and therefore on the overall success of the system.

11.4.2.3 The Case Base Container

The need for changing a case base mainly results from outside demands. If the context changes, cases have to be added, removed or modified.

There are different operators for modifying a case base. They are connected to building a case base for the first time and to the observations made for the learning algorithms IB1, IB2, and IB3. For the major operators we refer to Chap. 10. They are in short:

- Adding and deleting a case.
- Specialising a case: Adds a variable to the problem formulation to restrict the applications of the solution. This will be formulated in terms of constraints.
- Generalizing a case: Removes a variable from the problem formulation or a precondition for a rule to extend the applications of the solution.
- Modifying a case: A combination of the two operators above that changes the value of an attribute.
- Alter a case: Remove a variable from an attribute and add it to another attribute.
- Cross-cases: Convert two cases with equal solution attributes into one.

A danger arises if the case base becomes too large. Then maintenance can get difficult and retrieval time may increase.

11.4.2.4 The Similarity Container

The simplest view is on similarity relations where user feedback is available. This is more involved for measures. If we restrict ourselves to weighted linear measures, the major change actions are:

- Change a weight.
- Change a local measure. This has again little influence on other system parts. It is usually a consequence of changing the domains of attributes.
- Extend a measure to a new attribute: These are changes subsequent to the changes of the vocabulary.

This is discussed in Chaps. 6 and 7 on similarity. All these changes may influence the overall performance of the system. This, however, will have a great impact on the retrieval phase in the process model.

11.4.2.5 The Adaptation Container

Adaptation changes may greatly influence the overall performance of the system. Changes in the adaptation may exclude for instance the medical treatment in the therapy.

Rules and adaptations can be added, deleted and modified in an arbitrary way. They affect:

- The case base, because cases may become redundant or missing.
- The retrieval because the presence of adaptations may increase its difficulty.
- The user-friendliness: Rules are more compact and easier to overlook by the customer.

The motivation for the changes results from adding or removing cases or from the possibility or impossibility of using adaptation methods. It can improve the reuse phase in the CBR process model. The treatment of rules is very knowledge-intensive.

11.4.2.6 Provenance

The cases can come from import or modification of other cases from outside or from applying adaptation. In Chap. 12, Advanced CBR Elements, we consider provenance. In that chapter, we emphasize that the pattern records on how the cases are generated play a role. If these patterns are recorded then they can be used for the generation of new cases requested for maintenance. There we discussed this from the perspective of trustworthiness. For maintenance, however, there is an additional point.

Formally, we define:

Definition 11.1 A case history of a case is an action sequence of operators that describe what happened to the case:

$$\text{history}(\text{case}) = \text{case}_1, \text{case}_2, \dots, \text{case}_n$$

$$\xrightarrow{\text{op}_1} \qquad \qquad \xrightarrow{\text{op}_{n-1}}$$

where

- (i) Each operator is an internal or external operator
- (ii) case_1 is an imported case
- (iii) $\text{case}_n = \text{case}$

A provenance record is a set of case histories. Provenance records are or should be stored in the experience factory. In practice, the records will not be assigned to a single case but rather to a set of cases that share a common history.

The use of these records is twofold. The first use is concerned with reliability. The correctness of the operator applications can be controlled directly and will not be discussed here. The ultimate origin is the source from which the case is imported. Here, two aspects have to be viewed:

- (a) The reliability of the source.
- (b) The context in which the source was used.

For the second point, the major impact is that even if the case was correct and delivered perfect solutions and for the goal actually intended, that fact may not anymore be valid. Therefore, the source should at least have the following information units: A time stamp and a goal description.

Take, for instance, a case that depended on some regulation that has a time stamp d . If d was a long time ago, the case may not be valid anymore even if there was no change reported. The second use is for maintenance purposes and it is again twofold. Firstly, if some case needs to be changed then the history can support this by derivational analogy. Secondly, it can help to detect the reasons for weaknesses by partially replacing feedback if this is not available. For instance, unreliable sources may be replaced by better ones.

11.4.3 Systematic System Maintenance

Because maintenance is part of the overall business process of a company, it requires professional organisation with a well-defined structure. This will be discussed in the same way as was done for development. The most prominent maintenance operations are concerned with the cases and the case base and we will concentrate on them. Maintenance requires some additional techniques from outside CBR in order to be performed well. An important part of the organisational structure is change management. The basis for maintenance was provided in Chap. 10, Evaluation, Revision, and Learning.

11.4.3.1 Experience Factory

The experience factory not only is related to system development but is also relevant to maintenance. It contains examples and guidelines about when and in what ways maintenance operations (including evaluations) should be performed. In addition, historical records of the origins of the cases are quite useful for maintenance. Therefore, the experience factory has always to be updated.

11.4.3.2 Change Management

Essential to maintenance is reacting to changes to the knowledge environment. If the changes are known then one needs a way to propagate them to the place where they are used. To organise this is the purpose of change management.

One describes the actual change actions by operators. These operators are defined on information units, for instance, on attributes or cases. Operators are defined on a general level and can be instantiated.

If an incoming change or information unit is reported, then the change rules are describing the subsequent changes. The arrival of new information is called an event and the question is, who has to be informed about the event? Such an event has consequences on a few parts of the system only. To regulate this, one formulates some conditions. Events and conditions are the preconditions of the rules. If they are satisfied, an action takes place. These rules are called Event-Condition-Action (ECA) rules. An ECA rule references an *Event*, refers to a *Condition* and calls an *Action*.

The actions are subclasses or instances of an Action Class. The Action Class describes the executable actions. An important example is a notify action. This action notifies some agents about an event and the conditions describe the types of addressed agents.

Example Changes in the allergy list (the document “aller”) cause changes in the therapy list (the document “ther”). A short form of the rule pattern is:

Event:

- REPLACE(aller, therapy, patient, allergy x , allergy y)

Condition:

- Patient is in aller and in ther

Action:

- NOTIFY(responsible(ther), “event”, allergy of patient, alternatives, allergy manager)

This is easily understandable without further explanation. An instance of the pattern is given by:

aller = all1, ther = th5, patient = Maria, allergy x = no, allergy y = yes, responsible(ther1) = Pedro, alternatives = medication, allergy manager = Fabio.

In this case Pedro is informed by Fabio about Maria’s allergies.

Because some solutions will refer to the therapy list, some cases have to be changed. If this change avoids serious consequences it has to be done immediately.

Rule patterns for actions are defined at compile time; they represent general knowledge about the company and the processes. The generality is contained in the variables that can be instantiated in their domain. Patterns use typed variables (e.g., for products, documents, agents). Instances of rules are generated at runtime and this is again triggered by events. The time of generation of rules is not necessarily identical with the time they are used. This time is determined by the event that is part of the precondition of the rule.

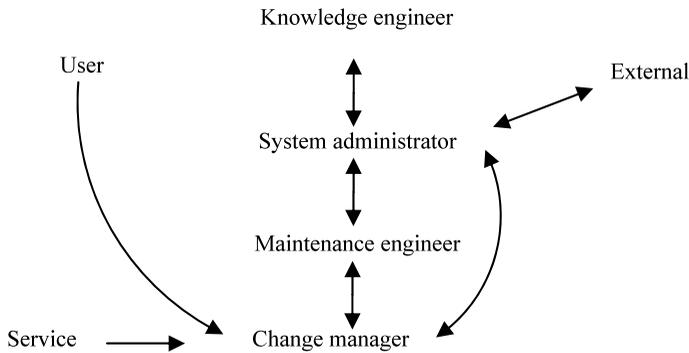


Fig. 11.9 The maintenance team

The executions of the actions, for instance, a notification, have again to be formalized such that an automatic execution is possible. This could be the sending of an email, which can easily be automated.

Rules have to be organised in a rule system. A basic structural element is:

Definition 11.2 An organisational concept E has a change impact on a concept F if a change in E results in a change in F .

For example, a list describing the ingredients of a medicine has a change impact on the list describing the doses. Change rules have the special property that they have to be executed, while rules in general can be executed optionally.

11.4.3.3 Maintenance Objects

Maintenance takes place on three levels.

- The top level considers the model. This includes mainly the vocabulary, the similarity measure, the rules, and the retrieval functions.
- The intermediate level considers the case instances:
Generate, delete, and adapt cases.
- The lowest level considers the implementation.

11.4.3.4 Maintenance Participants

The major participants (agents) in the maintenance task are shown in Fig. 11.9. This is quite similar to the development team, and usually both overlap. User and Service are outside institutions that provide information concerned with the available solutions. In fact, they are usually the only parts of the environment that can be contacted. The change manager accepts change results and distributes them to the responsible agents.

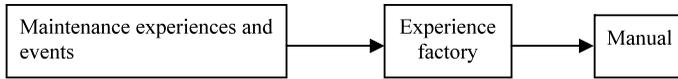


Fig. 11.10 Manual development

The administrator should take care of a maintenance manual associated with the experience factory. The manual will be developed over time, as shown in Fig. 11.10.

11.4.3.5 Testing and Measuring

The purpose of these activities is to detect failures and weaknesses of the system.

The change management collects:

- Information from Service
- Information and feedback from User
- Information from the maintenance engineer about system changes

The change management reports either continuously or at certain time points. A company usually has a hotline that accepts complaints and information from the users. These reports can contain information about maintenance indicators such as failures, errors, poor performance or inconvenient handling.

The information from User and Service are of very different nature; the latter changes the underlying model of the CBR system.

Service reports on changed objects used for the solution. This includes available products, methods, techniques, and so on. This results in:

- Changing a name.
- Adding, deleting or changing whole objects and their properties. It needs to be propagated to the other containers. In addition, it can affect the weights and hence the similarity measure.
- Changing the structure of objects. This is concerned with more radical changes and has also to be reported to the other containers.

The reports from users are concerned with the use of the system. They apply to the CBR system, reports on errors, requests, and new requirements, and suggest cases or other changes. In addition, users may complain about poor performance and deficits in the periphery. The user is the major source for information about system behaviour.

The results from the reports give rise to testing and measuring operations. These methods should be stored in and made accessible from the experience factory.

11.4.3.6 Triggers

The maintenance engineer decides about possible tests on the basis of maintenance indicators. Measurements as well as maintenance actions need to be triggered in

Table 11.3 Change dependencies

System parts	Change effort	Dependencies
Vocabulary	High	High
Similarity measure	High	Low
Case base	Low	Differs
Rules	High	Low
Periphery	Differs	Low

order to determine when they should be performed. There are three main types of triggering:

- Event-based.
- Ad hoc.
- Periodic.

The event-based give rise to the application of ECA rules. Ad hoc means there is no systematic method.

Periodic refers to fixed time intervals. This is copied from the maintenance of technical devices. It relies on experiences that determine in which intervals measurement and maintenance are useful and necessary. For the periods, there are guidelines in the experience factory. The system administrator and the company management should regulate this in a very well-organised way.

11.4.3.7 The Periphery

The periphery is subject to maintenance because sometimes it has to be changed and updated too. In general, changing or updating the periphery does not affect the other parts of the system but can influence user-friendliness. For that reason, maintenance is applied for additional new user types. For example, if the old system was dealing with experts only and the new users are beginners, they need a different user interface. This is an event-based triggering.

11.4.3.8 Dependencies

For change management it is important to be aware of dependencies. This means: If one part is changed, what other parts may be affected in what way, and who has to be informed? This depends on the special application; nevertheless, some general indications can be given as shown in Table 11.3.

11.5 Tools

The tool jColibri (Recio-García et al. 2005) provides support for development as well as for maintenance. For supporting maintenance in general, see, for example,

the SIAM shell is designed to support maintenance (Roth-Berghofer 2002). However, presently there is no tool available for the entire maintenance process.

Some CBR systems support connections to a database by offering an intermediate language. An example is CQL2, the case query language of the CBRWorks (<http://cbr-works.net>).

Zaluski et al. (2003) describes a semiautomatic tool for extracting knowledge from texts and documents and transferring it to cases.

11.6 Chapter Summary

In the first part of this chapter the systematic development of a CBR system was investigated while the second part was concerned with maintenance. Besides general advice from software development, the filling of the knowledge containers as a CBR-specific element was discussed. This has governed development as well as maintenance. The treatment of the individual containers provides a local view. This is complemented by a global view that considers the overall performance of the system. For both, development and maintenance, we also discussed professional organisational structures. The role of the participating agents was analysed. The individual steps are integrated into the systematic organisation. Because maintenance is an integral part of a system, the CBR cycle was extended to include it. Maintenance includes test as well as maintenance operations.

11.7 Background Information

Up to circa 1995, development and maintenance were more an art than a science. For development, no systematic regulations and advice were known. The need for maintenance as well as systematic development did not come up in many situations. In industry, maintenance could only be justified if the systems were used for a long period of time in which the context was changing.

The most influential project for systematic development was originated by the European projects INRECA I/II. A general description is given in Bergmann et al. (1999) and Bergmann (2001). It gives guidelines for the development of general systems although the emphasis is on call centre applications. The INRECA system also integrates CBR and decision trees. The basic common data structure is the tree, used for decision trees as well as kd-trees for retrieval.

An overview of general software metrics is given in Catal and Diri (2008). The experience factory (EF) concept was introduced by Basili (1995). For ISO norms, consult Kehoe and Jarvis (1996). For the concept of measurable organisational value for evaluating the benefit of software development projects, see Marchewka (2009).

EF can be used as an organisational infrastructure (i.e., roles, responsibilities, processes, organisational implementation and management strategies, competence development strategies) for a CBR system (Althoff and Wilke 1997). A detailed

description about a formalism for software engineering ontologies in an experience factory is given in Tautz and Gresse von Wangenheim (1998).

Basic structures for maintenance have been introduced in Leake and Wilson (1998). There are three dissertations on maintenance that contributed to systematic maintenance, Wilson (2001), Roth-Berghofer (2002), and Iglezakis (2004). In Roth-Berghofer (2002), a methodology and a (quite complete) suggestion for a manual can be found, where the SIAM methodology is developed. An overview is also given in Leake et al. (2001). Cummins and Bridge (2009) report on editing systems like ICF and RENN. For provenance, see Leake and Whitehead (2007).

11.8 Exercises

Exercise 1 Suppose you have a CBR system that allows retrieval of cases and queries step by step from a database. Extend the system in such a way that a whole set of queries and answers can be retrieved by one command so that the results can be evaluated.

Exercise 2 Describe change rules that take care of changes of available databases used by a CBR system.

Exercise 3 Develop a business model that specifies costs for maintenance and bad performance and tries to balance them.

Exercise 4 For a domain of your interest describe the data together with their histories.

References

- Althoff K-D, Wilke W (1997) Potential uses of case-based reasoning in experience based construction of software systems and business process support. In: Bergmann R, Wilke W (eds) GWCBR'97: sixth German workshop on case-based reasoning: foundations, systems, and applications. Technical report LSA-97-01E, Centre for learning systems and applications, University of Kaiserslautern, p 31
- Basili VR (1995) The experience factory and its relationship to other quality approaches. *Adv Comput* 41:65–82
- Bergmann R (2001) Highlights of the European INRECA projects. In: Aha DW, Watson ID (eds) ICCBR 2001: case-based reasoning research and development. 4th international conference on case-based reasoning, Vancouver, BC, Canada, July/August 2001. Lecture notes in computer science (lecture notes in artificial intelligence), vol 2080. Springer, Berlin, p 1
- Bergmann R, Breen S, Göker M et al (1999) Developing case-based reasoning applications: the INRECA methodology. Lecture notes in computer science, vol 1612. Springer, Berlin
- Catal C, Diri B (2008) A systematic review of software fault prediction studies. *Expert Syst Appl* 36:7346–7354

- Cummins LL, Bridge DG (2009) Maintenance by a committee of experts. The MACE approach to case-base maintenance. In: McGinty L, Wilson DC (eds) ICCBR 2009: case-based reasoning research and development. 8th international conference on case-based reasoning, Seattle, WA, July 2009. Lecture notes in computer science (lecture notes in artificial intelligence), vol 5650. Springer, Berlin, p 120
- Iglezakis I (2004) Case-base maintenance of case-based reasoning systems in classification domains. Dissertation, University of Würzburg
- Kehoe R, Jarvis A (1996) ISO 9000-3: a tool for software product and process improvement. Springer, New York
- Leake DB, Whitehead M (2007) Case provenance: the value of remembering case sources. In: Weber RO, Richter MM (eds) ICCBR 2007: case-based reasoning research and development. 7th international conference on case-based reasoning, Belfast, UK, August 2007. Lecture notes in computer science (lecture notes in artificial intelligence), vol 4626. Springer, Berlin, p 194
- Leake DB, Wilson DC (1998) Categorizing case-based maintenance. In: Smyth B, Cunningham P (eds) EWCBR-98: advances in case-based reasoning. 4th European workshop on case-based reasoning, Dublin, Ireland, September 1998. Lecture notes in computer science (lecture notes in artificial intelligence), vol 1488. Springer, Berlin, p 196
- Leake DB, Smyth B, Wilson DC, Yang Q (2001) Introduction to the special issue on maintaining case-based reasoning systems. *Comput Intell* 17(2):193–195
- Marchewka JT (2009) Information technology project management. Wiley, Hoboken
- Recio-García JA, Sánchez-Ruiz AA, Díaz-Agudo B, González-Calero PA (2005) jColibri 1.0 in a nutshell: a software tool for designing CBR systems. In: Petridis M (ed) 10th UK workshop on case based reasoning, Peterhouse College, Cambridge, 12th December 2005. CMS Press, Greenwich
- Roth-Berghofer TR (2002) Knowledge maintenance of case-based reasoning systems: the SIAM methodology. Dissertation, University of Kaiserslautern
- Tautz C, Gresse von Wangenheim C (1998) REFSENO: a representation formalism for software engineering ontologies. Technical report 015.98/E, Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern
- Wilson DC (2001) Case-based maintenance: the husbandry of experience. Dissertation, Indiana University
- Zaluski M, Japkowicz N, Matwin S (2003) Case authoring from text and historical experiences. In: Xiang Y, Chaib-draa B (eds) AI 2003: advances in artificial intelligence. 16th conference of the Canadian society for computational studies of intelligence, Halifax, Canada, 11–13 June 2003. Lecture notes in computer science (lecture notes in artificial intelligence), vol 2671. Springer, Berlin, p 996

Part III

Advanced Elements

This part extends Part II in a way that the topics presented there are enriched by concepts and methods to a more advanced level. This level is intended for readers who either want to get a deeper insight or who deal with special applications where the material can be of use. The topics are somewhat heterogeneous and the readers or lecturers can select them depending on their interest.

Chapter 12 continues the study of basic elements but on an advanced level. First the knowledge containers are investigated more closely. Of specific interest is shifting knowledge from one container to another one. With respect to knowledge representation ontologies are discussed. Much emphasis is put on contexts where different context levels are introduced. Then the study of cases, case bases and whole systems is extended, including distributed case bases, correctness, and provenance. Finally the relation to analogy is discussed.

Chapter 13 extends similarity. Here no new measures are introduced but rather background material is given. The semantics of similarity measures is put on a formal basis by relating to the utility concept. This leads to a new discussion of the axioms for the measures. The formal view is complemented by looking at subjectivity. Then non-functional aspects of the measures are considered. Finally some relations of similarity to other topics like explanation and logic are sketched.

In Chap. 14 first two advanced retrieval methods are introduced, Case Retrieval Nets, and Fish and Shrink. As a general extension to retrieval, fuzzy retrieval is considered. For improving, retrieval preprocessing and footprints are introduced. An important issue for e-commerce is diversity that deals with the similarity of several nearest neighbors to each other.

Chapters 15 and 16 deal with uncertainty. Chapter 15 is concerned with general aspects of uncertainty. It includes rough sets, fuzzy theory and possibility theory. Detailed relations of these concepts to similarity measures are provided. Chapter 16 discusses the relations between probability and CBR. In particular, similarity measures for random variables are investigated. At the end there are short introductions to Bayesian Reasoning and Stochastic Processes as far as they are related to CBR.

Chapter 12

Advanced CBR Elements

12.1 About This Chapter

In this chapter, we move towards more advanced aspects of CBR design, development and deployment. We start by discussing an advanced aspect of containers, of their relationships. The new concepts presented here are however of heterogeneous character in order to cover some missing aspects. We also present a deeper discussion of contexts; CBR systems, their properties, their conditions, and so on; and ontologies.

12.2 Discussion of the Relations Between Containers

By now the reader should be quite familiar with CBR. Consequently, it may be time to start shifting things around and to challenge the reader. Right now we envision CBR as a methodology that implements a process starting from a new problem, uses this new problem to guide retrieval of a case with a similar problem, reuses its solution, revises it, and possibly learns something. We have previously introduced the issues with respect to similarity as a proxy to usefulness that is formulated for each system based on required knowledge.

Figure 12.1 shows the four containers for a given system. The vocabulary container has some knowledge. The similarity container has more knowledge, so the system can successfully find similar, that is, useful, cases. The case base has several cases. The adaptation container has knowledge to adapt cases that are retrieved based on the knowledge of the similarity container.

When we talk about knowledge in a container we are mainly concerned with additional knowledge for problem solving rather than the knowledge in the basic definitions.

Note that similarity is associated with retrieval while adaptation is associated with reuse. Note also that retrieve and reuse are adjacent tasks in the CBR methodology. Now consider removing most of the knowledge from the similarity container. We could do this by taking each candidate case from the case base and subject

Fig. 12.1 The four containers in a given system

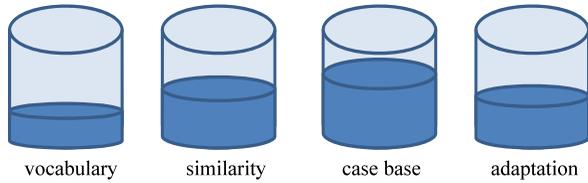
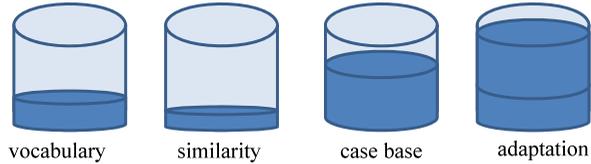


Fig. 12.2 The possible four containers in a given system



it directly to the adaptation container, in order to determine how much adaptation would be required for each candidate case. With this information, the case to be reused would be the one with fewest adaptation needs. This is a method known as adaptation-guided retrieval. It shows that containers in CBR are strongly related and may be able to compensate for weaknesses in other containers. Figure 12.2 depicts the containers in such a situation. As less knowledge is available in the similarity container, additional knowledge is required in the adaptation container in compensation.

There are several relations between the containers. A simple example is that one can omit cases from the case base by adding a rule in the adaptation container that generates them. Another example is between the vocabulary and the similarity containers. Imagine cases describing cars at the specific level of brands. The similarity container uses knowledge about their categories in order to determine that, for instance, both Infiniti FX35 and BMW X5 are luxury sports utility vehicles. Rather, the vocabulary could indicate directly in cases that Infiniti FX35 is a luxury sports utility vehicle and thus this knowledge would not be needed in the similarity container.

An observation is that any of the four containers can in principle contain essentially all relevant solution knowledge. In practice, however, this is not advisable and usually is not even of theoretical relevance. We describe the content of the containers in an ideal situation.

The ideal vocabulary: We need for each problem description p simply an additional (ideal) attribute that gives the correct solution.

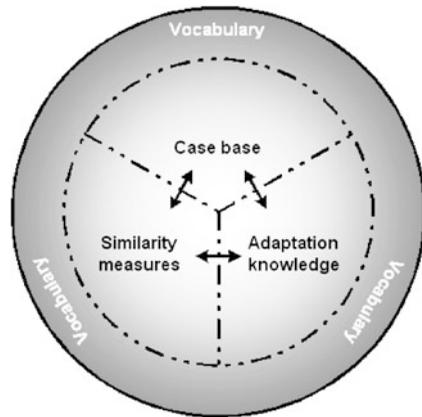
The ideal similarity measure: For an ideal measure $\text{sim}_{\text{ideal}}$ we could have $\text{sim}_{\text{ideal}}(q, p) = 1$ if the case (p, s) provides some best solution s and $\text{sim}_{\text{ideal}}(q, p) = 0$ otherwise.

The ideal case base: An ideal case base in this sense would simply contain all possible cases.

The ideal knowledge in the adaptation container: One could simply ignore the cases and construct the solution from scratch using the adaptation rules.

Such *ideal* objects are of course not available. However, one can see that in principle each container can essentially carry all knowledge needed for problem solving.

Fig. 12.3 Shifting



This suggests also that shifting some knowledge from one container to another one can change the problem-solving capacity.

The answer to the question about what knowledge goes in which container and in what way depends to a large degree on the way the knowledge is made available and presented. In the development of a CBR system one should look at filling containers as an incremental process where knowledge may be moved from one container to another. Often one starts with a given case base and a simple measure. Later on some cases may be deleted because of duplicates, a more intelligent measure and a useful adaptation rule. In Chap. 11, Development and Maintenance, this process is studied in detail and from a methodological point of view.

To discuss the filling procedure it is advisable to distinguish between *design time* and *runtime*, borrowed from traditional programming terminology. By design time or compile-time we mean the time when a system is built, in particular, before any actual problem is presented. By runtime we mean the time when an actual problem is presented and solved. Of course, system development and problem solving can be interleaved.

The first three containers are typically filled at design time; they contain compiled knowledge. On the other hand the case base may be filled incrementally. An important point is that the cases do not have to be understood at design time. The cases can be *stored* in the case base without our understanding them at all and can simply be copied from existing files. This simplifies the traditional bottleneck of knowledge acquisition for knowledge-based systems to a large degree.

For all other containers, one has to *understand*, in some sense, the knowledge:

- One has to collect all necessary and useful attributes.
- One has to define suitable similarity measures.
- One has to define useful and correct adaptation methods.

There is an interaction between the containers. As mentioned, knowledge can be shifted between them as shown in Fig. 12.3.

Figure 12.3 shows some of the relations between the containers. Many details are discussed in Chaps. 6 and 7 on similarity, Chap. 9 on adaptation, and Chap. 10

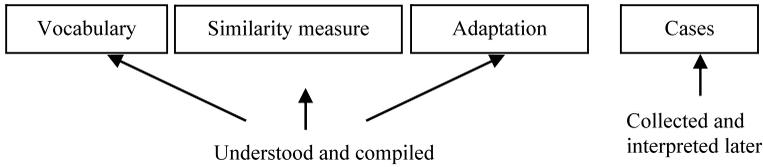
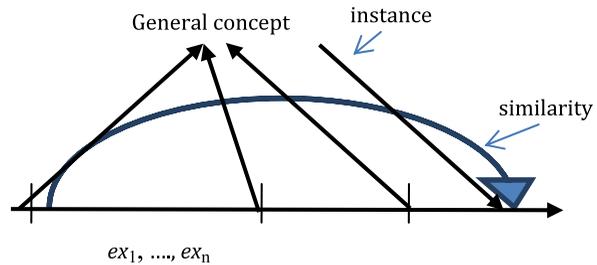


Fig. 12.4 Processing of CBR knowledge containers

Fig. 12.5 Eager versus lazy



on learning. The interaction means that knowledge units (in whatever form) can be shifted from one container to another. As illustrated in Fig. 12.4, only the vocabulary, similarity measure, and adaptation methods are processed as in other knowledge-based systems. The cases are collected and interpreted later.

We sketch and partly repeat the roles and the significance of the containers. More details will be given in later sections of this chapter.

The choice of the attributes is crucial for success. Some important quality criteria for the vocabulary container are:

- Independence of the attributes: Dependency creates problems for the similarity measures.
- Completeness of the attribute set: If an attribute is missing, one cannot deal with it.
- Minimality: Avoiding redundant attributes increases quality. Deleting an attribute should lead to a solution with lower quality.

The cases represent experiences from past episodes either as (problem, solution) pairs or simply as solutions. Cases occur also in machine learning as essential elements. If n examples ex_i are given, then

- (a) Machine learning creates a general concept (“eager learning”) that is instantiated for new problems.
- (b) CBR collects the examples in a case base, learning from user interaction (“lazy learning”). For a new problem, the nearest neighbour delivers the solution. This is illustrated in Fig. 12.5.

Figure 12.5 shows the difference between general machine learning as concept learning and CBR, as already introduced in Chap. 10, Evaluation, Revision, and Learning. Machine learning generates immediately a result, namely, a general concept. For a new problem, this returns the solution as an instance or a generalization

of the learned concept. Similarity returns for each individual problem the nearest neighbour from which a solution is obtained.

One cannot say that one of the views is superior in general. This depends on the problem situation and the context. If we need and can obtain a reliable generalization that we use often, then eager methods are good. On the other hand, obtaining generalizations can be rather time consuming and is limited by a lack of formalized descriptions. If we rarely use it then the effort does not give sufficient payback.

There are essentially three such sub-containers:

- (a) The *adaptation container*, which holds the proper adaptation rules.
- (b) Rules that draw conclusions from the query or the solutions. Such rules occur in any knowledge-based system. These conclusions complete the queries as well as the conclusion and therefore influence the case container.
- (c) Rules that change the similarity measure. This means they change, in particular, the importance of attributes. For instance, “IF brand (car) = Ferrari THEN price less important”, which will be discussed in Chap. 13, Advanced Similarity Topics.

When one starts to develop a CBR system, initially all containers are empty. The development results essentially in filling the containers and an inadequately filled container may prove to be a burden to other containers.

In order to fill the containers, certain knowledge has to be available. The process of making it available is called *knowledge acquisition*. The necessary effort differs among the containers. The term *inadequately filled* refers to deficiencies in a knowledge container that affect quality aspects of a CBR system, such as correctness of solutions, competence of the system, and efficiency of problem solving or maintenance aspects.

One distinguishes between knowledge-poor and knowledge-intensive systems. Knowledge-poor systems require little knowledge besides that of cases about the problem area. The case representation is usually a flat vector of attribute values, and adaptation, if present at all, is of simple character. The similarity measures are usually quite simple. For knowledge-intensive systems, all steps and all knowledge containers require background knowledge about the target problem area.

This can also be expressed in the following way:

- Data-intensive is knowledge-poor and a case is essentially a data record; one bases the similarity on a simple metric.
- Knowledge-intensive is data-poor and a case is a more or less complex user experience; the similarity is some kind of an explanation process.

12.3 Contexts

12.3.1 Generalities

In principle, there is usually an infinite amount of knowledge that has a relation to a problem of interest (except a purely combinatorial one). What one actually needs

depends on the part of the world that is relevant to the problem. This is called the relevant context or simply the *context*. This context contains everything of interest to the problem, for instance, the goals, the costs, the constraints, or the exceptional situations. In particular, the context specifies the intended utility, so a lot can be obtained from users. There are various ways a goal can be missed, for instance, if the knowledge is incomplete, too general, confusing, or not understandable.

One or more contexts are described in a systematic way, as shown next. Each concrete context can use an ontology for its description.

12.3.2 Different Contexts

We distinguish external and internal contexts. The external context considers what happens around the performing agent (e.g., the specific task, outside events, and general circumstances). The outside events determine, in particular, what unexpectedly occurred. Such events may occur more or less often and they can be recorded as *effects*.

The internal context is concerned with the knowledge and experience of the agent. To obtain this knowledge, a communication with this agent is necessary. This context is usually for a particular unknown agent and one needs information from the outside. One purpose of a dialogue is to obtain such knowledge.

In order to deal with different contexts properly we introduce a partial ordering on the set of contexts. The ordering says that one context can be more or less general than another one. We start with some definitions.

Definition 12.1

- (i) A knowledge or information unit is a primitive concept. It may be represented in a formal or informal way.
- (ii) A context is an (abstract) set C of knowledge units. This means, it may not be represented in a specific way and the representation may even not be known.
- (iii) A context C_1 is more specific than a context C_2 (or C_2 is more general than C_1) for a term T (a concept, an action, an object, and so on) if the term T is less ambiguously described in C_1 than in C_2 . Then T allows fewer interpretations in C_1 than in C_2 .

Notation: $C_1 \geq_T C_2$.

This means that in a more specific context more knowledge is stored that restricts the possible interpretations of the term. This makes the set \underline{C} of sets of knowledge units a partially ordered set (\underline{C}, \geq) with respect to a knowledge unit term T . As an example, the context containing just *planning* is more general than the context *travel planning*, and this is again more general than the context describing *my next trip*.

The more general a context is, the less specific is the knowledge described. On the other hand, the less generally a context is described the more easily it can be represented and retrieved. An ideal context for a set of problems is one that contains exactly the knowledge needed for these problems.

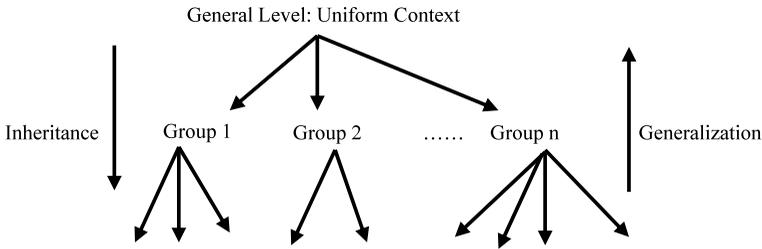


Fig. 12.6 Context levels

To simplify the situation for practical purposes we introduce levels of contexts and define three context levels as shown in Fig. 12.6. These levels are concerned with agents that need context information for certain processes:

- (1) The general level: Everybody uses the terms in the same way, for instance, when one searches for the Lufthansa schedule. That means, no more specific context can answer the question better.
- (2) The group level: There are groups of users and each group has a specific context that may differ from group to group. For instance, different social groups look for different activities for entertainment. Another example is a diet plan in a hospital. It depends on the ailment of the patient, such as diabetes or a liver disease.
- (3) The individual level: The context depends on the needs of the specific user, for instance, when one searches for an employee with specific abilities or for a very special product to buy.

Part of the context is the utility of the user. Ordinarily, everybody has utility aspects from all three levels. The utility is partially individual, influenced by one or more groups, but shares also some general views. On the general level, one finds mathematical utility functions; the more one gets more specific, the more subjective the utilities that play a role. There are two major problems associated with contexts; both give rise to maintenance; see Chap. 11, Development and Maintenance:

- (1) These contexts are not static; they change over time. The speed of change increases the lower the level; on the general level, the change proceeds relatively slowly. Each change has to be reflected in a system that provides knowledge; this is known as a maintenance task.
- (2) The contexts are partially unknown. This is a problem not only on the general level. On the group level it may sometimes require, depending on the group, substantial effort to acquire. On the individual level there are mainly two ways. The first one is to learn preferences from user histories, which necessitates that these be recorded. The second way is direct communication with the user, where conversational CBR can be used.

At the general level, general search machines and retrieval techniques are located, such as those that search the Web. These are hard to beat for this purpose. The more one goes down the level hierarchy, machines mostly do not deliver what one is interested in. On the group level, CBR techniques are serious competitors. The recent ac-

tivities on the lowest level are called *personalization* and *context awareness*. These activities are not only task-centric but also user-centric. CBR techniques play an important role in those. This is reflected by many successful case-based recommender systems.

When building a CBR system one should investigate the necessary context as one of the first steps. It influences several other decisions, such as about the terminology used to address the user. This was discussed in Chap. 11, Development and Maintenance.

12.3.3 Contexts and Knowledge Containers

The contexts have a serious influence on the structure and the use of a CBR system, and affect all knowledge containers.

12.3.3.1 The Vocabulary Container

The context determines what terms and concepts are understood, common and acceptable. In the context of some company, certain terms may have a specific meaning, as in a bank. Also, certain methods may be common to everybody inside but unknown outside the company. A lesson is that in some contexts, such as political or religious ones, certain names or expressions have to be avoided.

12.3.3.2 The Similarity Container

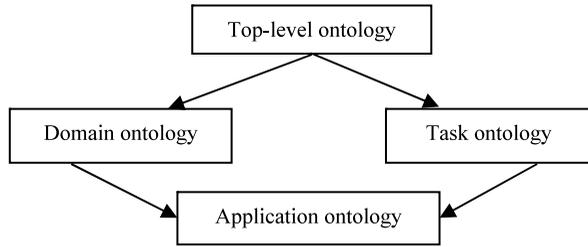
Because the context determines the utility, and a similarity measure should approximate it, there is a clear influence visible. This is because the preference relations of the user (which determine the utility and the similarity measure) depend on the context. A requirement is that slight changes in contexts should result in small changes in the measure too.

12.3.3.3 The Case Base Container

The case base contains the available solutions. This means that the case base does not only depend on the domain but rather on the context that defines the use. Certain solutions may be preferred, allowed, rejected or forbidden in some contexts. An example is in chain stores like supermarkets, where different customer types or local government and tax regulations for selling products define different solutions.

12.3.3.4 The Adaptation Container

It can happen that certain adaptation methods are simply not available in some contexts. It also occurs frequently that the availability of adaptation methods can change dynamically as a result of a changing context.

Fig. 12.7 Ontology levels

12.4 Ontologies

An ontology represents declarative and structural knowledge. This means that it can represent entities, what they are and how they relate to other entities in a given scope.

The relationships between entities in ontologies are organised hierarchically. Ontologies are rich representations because they rely on logic differencing. They entail a great variety of information about the entities, such as types, constraints, cardinalities, semantic entailments, functions, and axioms. The relationships between entities can be formally defined in several ways, namely, as inclusion, spatial, cause-effect, rationale, means-end, sequence, attribution, and so on. A typical ontology is shown in Fig. 7.2 of Chap. 7, Complex Similarity Topics.

The knowledge comes usually from different sources that have different representation formats as well as different semantics for the used terms. The idea of an ontology is to collect, combine and restructure such knowledge.

Major characteristics of an ontology are:

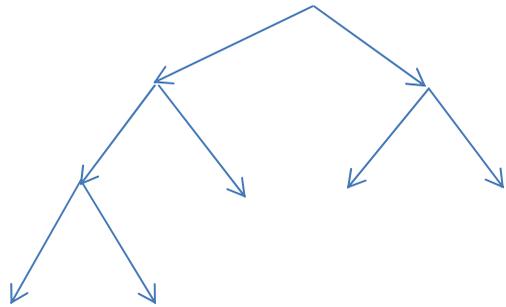
- formal: it is machine understandable.
- explicit specification: it contains explicitly defined concepts, properties, relations, functions, constraints, axioms, and so on.
- shared: it represents consensual knowledge of a community.
- conceptualization: it is an abstract model of some phenomenon in the world.

There are different types of ontologies because they can talk about different issues and different levels of abstraction as seen in Fig. 12.7.

The original motivation to build ontologies in artificial intelligence was to provide commonsense reasoning. For this reason, ontologies themselves need to have types and purposes. The most general ones, the top-level ontologies, are known as base or abstract ontologies. These conceptualize the most general entities such as air, mass, set, relation, gravity, and so on. Even for general entities that are common to all, ontology designers realised they needed to specialise; thus there are specific ontologies for time, transportation, and so on. Other main types are domain ontologies, application ontologies, and linguistic ontologies that support natural language. When using ontologies for an application, ideally one should append one top-level (base) ontology to all other required ontologies.

Task ontologies contain concepts and facts of interest for some application type as diagnosis ontologies, planning ontologies, and configuration ontologies. On the other hand, domain ontologies describe shared conceptualizations of experts in a

Fig. 12.8 Tree structures



field such as microbiology or software engineering. As mentioned earlier, task and domain ontologies are required when a specific application is built. The most specific ontologies are application-dependent; they are designed based on the task and the context (e.g., users and domain) in which they are meant to be used.

Entries in ontologies are usually structured in a directed graph, which is often a tree. We show a tree-style graph in Fig. 12.8.

As an example, we consider an ontology of transportation resources in Fig. 12.9. At the top are features applying to all the resources (e.g., capacity). On the second level the resources are categorized by terrain. On the third level are further variations; the fourth is the level of specific objects.

12.4.1 Ontologies in CBR

There are multiple ways to benefit from ontologies in CBR. The most thorough form describes cases and all other CBR processes as entities of an ontology. Another way is to use ontologies exclusively for case representation. Yet another way relies on ontologies exclusively for commonsense reasoning in support of processes such as similarity assessment. This last use, relying on ontologies for one aspect only, may also be exemplified by the use of linguistic ontologies for support of natural language needs.

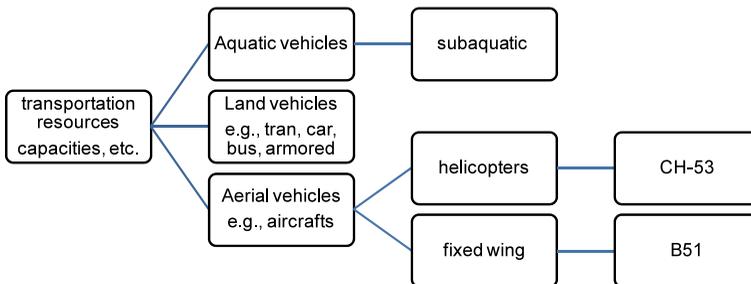


Fig. 12.9 Example tree

CBR ontologies can be used for all knowledge containers and steps in the process model. For the vocabulary, an ontology can be directly used in an object-oriented way. This supports building cases in the same way. For structural similarity measures, the ontology can help to define the measure using the taxonomic structures. For adaptation, the structure of the ontology supports the detection of rules for useful adaptations.

The relation between ontologies and CBR systems is that one builds or changes the systems when an ontology is already available. This raises the problem of using an ontology several times. Reuse of ontologies makes them valuable.

Ontologies may change over time. The change of an ontology can be regarded as a translation. A translation between ontologies makes the reuse of ontologies within the same representation language possible.

Example (for predicate logic):

- T_{old} is supposed to be an ontology for the organisation of personnel with a predicate $vacation(\text{time}, \text{location})$ (which shall express when and where) with corresponding regulations. Furthermore, let L_{new} be a language with the predicate $holidays(\text{time})$. A suitable translation is

$$t(\text{holidays}(x)) = \exists y(\text{vacation}(x, y)).$$

The knowledge base of T_{old} can now be reused.

For different logical languages, e.g., differently represented fragments of predicate logic, it is necessary:

- to understand the expressions of one language syntactically in the other language
- or
- to understand the expressions of both languages in a third language (i.e., exchange format).

The condition in order to achieve this is that the expressions are not ambiguous. Knowledge can be distributed over different ontologies:

- Different experts see and describe the world from different views; they have their own knowledge stored in different ontologies:
 - Financial experts, environment experts, regional experts (in particular, in global distribution), technical experts (again many possibilities), and so on.
- Each expert talks and thinks about the same real-world objects
 - in different terminologies
 - in different degrees of abstraction
 - in different forms of details
- In order to make use of such distributed knowledge, a principal problem of communication arises (see Chap. 22, Basic Formal Definitions and Methods).
- Some part of the problem can be solved by using normed terminologies.

The distribution occurs frequently in companies and we discuss this in Chap. 21, Knowledge Management.

Ontologies are used not only for describing knowledge containers but also for systematic context descriptions. Reuse of ontologies is of particular importance for them because they are changing constantly. This is shown in Sect. 12.3 on contexts.

12.5 CBR Systems

In this section we look at CBR systems from the viewpoint of representation and contexts. Here we pay attention to more or less general properties that a CBR system can benefit. Some properties are given from the problem and cannot be changed by the user. Others are wishes from the user. Both will influence the structure of the system and its ultimate success. Often, one can break down some of these properties to specific knowledge containers.

First we consider issues concerned with the system as a whole. In the subsequent section we pay attention to individual cases and case bases.

A basic question is about the wanted or unwanted properties that a CBR system may have. The term property is not well defined and has many interpretations. Here we understand such properties as demands or possible wishes of the user for the behaviour of the system. The main properties result from the fact that CBR is concerned with useful solutions of problems. This leads to the investigation of quality aspects. Quality is a property of the individual context because solutions are delivered in each context. This term is again not well defined and we discuss four main quality issues which can be influenced by all of the knowledge containers:

- a. Correctness and quality of solutions.
- b. Competence of the cases: Which problems can be attacked?
- c. Efficiency of problem solving.
- d. Are the solutions provided understandable by and usable to the user?

Correctness is traditionally a yes-or-no question in the sense of logic. However, often the question cannot be raised in this way. In complex situations one can measure correctness in degrees too. In general, the degree of demanded correctness depends on the applications. This allows correctness to be considered as a property. If one looks at the whole problem space, correctness and quality are distributed and refer to all knowledge containers.

Concerning any kind of quality, one can put different conditions on the case base; mainly:

- One wants a certain average quality.
- One wants a certain minimal quality.

This leads to specific development and maintenance tasks as discussed in Chaps. 10 and 11.

The next quality point is competence. This term is again not well defined and means how many of the intended problems the system is able to solve. Here we restrict ourselves to the case base and assume that the other containers are fixed.

The competence is roughly the ratio of the number of solvable problems to the number of possible problems. One observes that the competence is not uniformly distributed over the whole problem space; it is rather a *regional* property. The competence is high in some parts of the problem space and goes down in other regions. These regions do not have a sharp boundary. The reason for this is usually that the cases are collected for a certain problem-solving purpose but also used for problems where one has only some hope that the solutions can be useful or can be adapted properly. These topics have been investigated in Chap. 9, Adaptation.

The third quality aspect is efficiency. It is difficult to measure this in general but one can measure runtime efficiency. It is concerned with the retrieval of cases. This depends on the interplay between the case representation, the similarity measure and the retrieval algorithm itself. But this does not cover all elements connected with efficient problem solving. Here the situation is analogous to the question of competence; efficiency is distributed over the problem space. However, the efficiency regions are usually not the same as the competence regions. For this we refer you to Chaps. 8 and 14 on retrieval.

The problem of understandability starts with the query formulation. The user has to formulate the query in a specific prescribed format. Each such format requires certain knowledge in order that one can properly understand it. The presence of this knowledge depends on the user class. The same applies for the solution. The solution can be formulated and illustrated in many different ways. As a consequence, the system developer has to observe the relations between user classes and query and solution formulation. This needs user models, which is discussed in Sect. 12.5.2, Case Base Properties. The problems that arise from user-friendly representations that are not easily accepted by computers are discussed in Part IV. There we consider representations in the form of text, images and speech.

12.5.1 Case Properties

There are different kinds of properties of cases a CBR system can have:

- Neutral properties such as size or number of attributes.
- Properties pointing in a certain direction such as classification or planning.
- Properties pointing in the usefulness of certain tasks.

If a CBR system does not provide the demanded answer in a satisfactory way, this can have different reasons. Three major reasons related to case properties are:

- (1) The cases are corrupted. This means the cases are either incorrectly collected or they are noisy, such as when they contain wrong values.
- (2) Cases are badly selected or enter the case base by mistake.
- (3) The vocabulary is not well chosen. In this respect we have already discussed how the choice of the vocabulary is of importance for the formulation of both

the problem and the solution. In case the outcome is uncertain this should be mentioned in the solution. If the solution for a problem is equipped with a probability, the result is a pair (value, probability).

- (4) The similarity measure is not adequate.
- (5) There are retrieval errors. They can be the result of the strategy or the coding.

The discovery and corresponding repair are discussed in Chap. 11, Development and Maintenance. A problem comes from the fact that cases do not need to be understood at design time and therefore weaknesses are not discovered before runtime. The lesson here is that cases should come from reliable sources. This is the subject of Sect. 12.5.4 on provenance.

12.5.2 Case Base Properties

There are several requirements on case bases that need to be satisfied.

- The case base should only contain cases (P, S) , where the utility of S is maximal or at least very good for the problem P ; it may also contain cases labelled as negative containing unwanted solutions.
- There should be as many cases as possible in the CB because each additional case can possibly enlarge the competence of the system.
- The case base should be as small as possible because each new case can extend the search time.

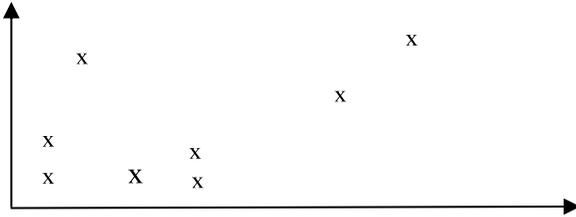
The last two requirements are in conflict and it depends on the problem area on how best to achieve a compromise. There are situations where having more than one case pointing to the same solution can be desirable.

A case base often exists a priori because one has recorded experiences. One should observe that the size of the case base is not a quality criterion. It is rather important that the base contain *good* and *useful* cases. Good means that the cases contain high quality solutions and useful means that it applies for many and difficult problem situations. One may think of the problem set as an ocean that has the cases as islands. If a large part of the ocean does not contain islands and other parts are packed with islands then one did not choose the island distribution well. This is illustrated in Fig. 12.10, where some areas are densely populated and some are almost empty, revealing a bad case distribution.

Such a case distribution has the consequence that some problems cannot be solved and for others there may be unnecessary cases. We tackled some related problems in Chap. 10, Evaluation, Revision, and Learning.

Next, we look at two properties of case bases that have a major influence on performance.

- Case bases can be homogeneous or heterogeneous. In a homogeneous base all cases are described by the same attributes, while this is not the case for a heterogeneous base.
- Cases may be episodic or prototypical.

Fig. 12.10 Case distributions

Homogeneous case bases require less effort for the definition of similarity measures. Heterogeneous information collections provide increasing difficulties. Such collections typically come from different sources using different vocabularies. To make things worse, these differences are often unknown. It requires much effort to cope with the situation. Sometimes it is advisable to split a heterogeneous case base into different case bases. However, this creates related problems.

Example Suppose there is a multinational travel agency distributed over the world. In different regions the local offices use different terminologies. In addition, they have different kinds of knowledge. Traveling in India is not the same as in Brazil. Here one has the possibility of splitting the travel office into parts where each part has its own consistent vocabulary and case base. However, this splitting is not enough. For each part one needs an agent who

- Understands its own case base;
- Can communicate with agents from the other parts.

These agents communicate in a specific terminology that is of interest to the whole company. Such splitting and communicating may also be of interest in intercultural situations.

Episodic cases are records of real events in the past. They are not systematically collected. They are just recorded as they come in. Prototypical cases are artificially constructed examples. They are in some sense abstractions of real events. One constructs them with the intention to cover the problems of interest in a systematic and efficient way. Prototypical cases are further discussed in Chap. 14, Advanced Retrieval. It is often the case that the two types of cases are mixed. One starts with episodic cases. If one discovers that certain problem areas are not well covered then other prototypical cases can be added.

12.5.3 Conditions

Conditions can be formulated in many ways. In a logical form they are called constraints; see Chap. 22, Basic Formal Definitions and Methods.

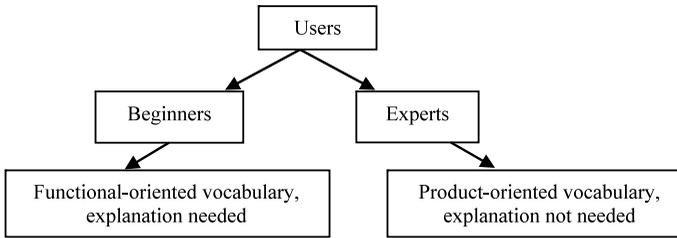


Fig. 12.11 User dependency

We distinguish between conditions on a case base and properties of a case base. Conditions on a case base are a consequence of the problem situation. If they are hard constraints then one has always to satisfy them.

Conditions on properties wanted by the user reflect demands. They are weak constraints and one can further weaken them. One judges them by the degree of success of the solution. That means the task is to choose an optimal CBR system for each special problem situation. For the investigation of conditions we introduce two taxonomies and distinguish between user-dependent and user-independent conditions. For the user-dependent conditions on wanted properties the nice situation is that there is a case base that contains experiences of interest to a large group of users in a way the group appreciates them. The more heterogeneous the group is, the more difficult it is to achieve our goal.

In the domain of technical products, a simple user-dependent structure is shown in Fig. 12.11.

This means that a novice understands more easily for what purpose the solution has to be used because that is the main concern of the user rather than to understand the structure of the solution. Suppose the problem is to select a machine to use. The selection requires knowing how one can use it, for what, and with what results. For an expert user the information about the products is mostly sufficient. Each user type gives rise to a number of demands on the CBR system, as terminology or user interface, but it does not affect the system's ability to solve the problem correctly. That means one wants:

- Pragmatic conditions.
- Representation conditions.
- Semantic conditions.

A user-dependent structure is shown in Fig. 12.12.

Here we observe an increasing amount of uncertainty when going from left to right. The problem situation has often a large impact on the representation form. Some domains like law require a textual representation while others need probabilities or exact numerical values.

Therefore, one has to describe the aspects representation, semantics, pragmatics, and finally user type. They generate conditions that influence each other.

Now that we have discussed conditions, we can return to the properties of the case base. They result from design decisions that influence how it functions. A case

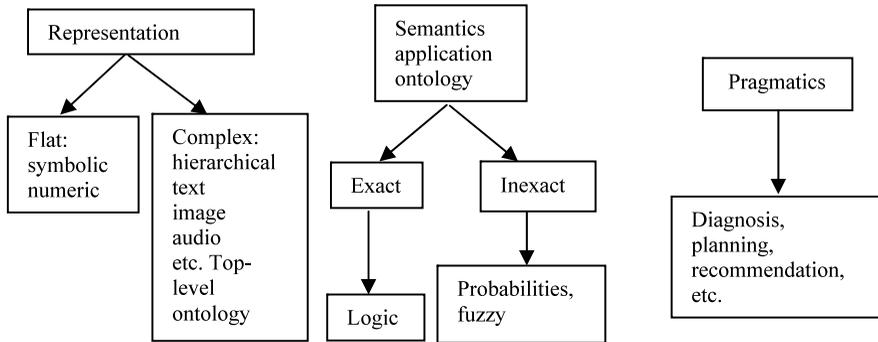


Fig. 12.12 User-independent conditions

base has some basic parameters:

- The number of cases,
- The number of attributes per case,
- The number of values per attribute.

These parameters cannot be discussed by just looking at them and by some counting. What matters are the relations between these parameters. One has to look at their problem-solving capacity. This again depends on demands on the solutions. If there are no demands then the parameters play no role. A useful kind of demand can be illustrated with similarity thresholds. This is simply a number α , $0 < \alpha < 1$, with the condition that the nearest neighbour has at least the similarity α . The case base parameters have to be discussed with respect to the threshold.

The study can be done in two views, a theoretical one and a practical one. The theory focuses on possibilities and limitations of the problem solving. A typical theoretical question would be, “Are there conditions on the parameters that certain problems can always or never be solved?”

A pragmatic point regards this with questions like, “Under what circumstances are what parameter relations recommended?”

12.5.4 Correctness and Provenance

With any kind of advice or presentation of a possible solution, the concept of correctness is associated. The difficulty is: How can we predict, determine or judge this? This plays a role in system development and in maintenance as well. In Chaps. 10 and 11 this is discussed.

In the logical sense, correctness is a yes-or-no question; see Chap. 22, Basic Formal Definitions and Methods. However, often it is measured in degrees. The pragmatics has an essential influence on the degree of required correctness of the solution. For instance, for weather prediction a lower correctness is more acceptable

than for surgery assistance. This means, in particular, that the minimal similarity threshold for a case to be accepted also depends on how crucial the problem is.

For many knowledge units, a correctness evaluation is often difficult; it is either a fact or it is not. For instance, if someone says, “my age is 18”, the person can present a passport. But how do we know that the passport is not fake? This leads to the investigation of the reliability of knowledge sources and provenance.

In the past decades provenance recording rarely took place in practical CBR. The reasons were probably the implicit assumption that cases are always correct and in the exceptional situations user feedback for corrections was available. This view has changed because of many situations where this did not take place. It should be mentioned that provenance recording is standard in economics, archives, and data mining.

Therefore, the question that comes up is, how much one can trust a CBR system? The trustworthiness in CBR results is closely related to the reliability of the sources of the cases, and of all knowledge containers. These sources of the cases may be more or less trustable. The investigation of the origin of the cases is more or less the only alternative to user feedback.

In most case bases the sources are not mentioned and therefore their reliability is not investigated. Provenance information can be given in different and more or less detailed ways. We distinguish two ways of getting information about the origin of the cases:

- (1) Internal provenance: Cases are generated during the system development or maintenance.
- (2) External provenance: Cases are imported from external sources.

During system development cases are not freely invented. They are imported or are modifications of existing cases. The first possibility is that some cases are modified in order to increase competence, which takes place in maintenance. The second possibility is that cases are generated by adaptation. In both situations the generation patterns are known; the question is only that they are mostly not recorded.

If the pattern is available, the correctness of the reasoning is a purely technical problem. The knowledge about such patterns is also useful for maintenance purposes when cases have to be modified again, for instance, because of a changed context. Ultimately, the investigation of the reasoning arrives at some case that is externally imported (unless the case is freely invented).

The first point of discussion is the reliability of the source itself. This cannot be judged from the CBR point of view. One needs a confirmation from outside, for instance, from (many) users who have employed the cases before.

The second point applies to cases that are totally correct in themselves. However, here the context and the precise problem of the application for which they have been built to be considered. The situation in the past may not be the same anymore and the problem may consider a slightly different goal. This can be discovered by a careful investigation of the origins of the cases.

This needs a systematic way of recording the origins. For this a set of descriptors should be used. These are essentially attributes with their domains that cover important aspects. Examples are:

- Type of recording: company records, public domain, patient history, interview, and so on.
- Temporal aspects: periods for data collections, when updated, and so on.
- Type of application: diagnosis, sale, planning, and so on.
- User references: used by whom, used how often, and so on.
- Reports: public, private, internal, and so on.

Such documentation can be stored in an experience factory that is updated regularly. Experience factories are discussed in Chap. 11, Development and Maintenance.

12.5.5 Distributed Case Bases

This section extends the discussion on heterogeneous databases. Sometimes a single case base is not very convenient or not available. This happens, in particular, if the solution space is distributed over several case bases for different reasons:

- Products are distributed over different electronic catalogues.
- Knowledge can be distributed over different knowledge bases.
- Different experts see and describe the world from different views.
- Each expert talks and thinks about the same real-world objects in different terminologies, with different degrees of abstraction, and with different form of details.

Distributed case bases occur naturally in multi-agent systems. There one has to deal with the problem of retrieving episodes that are themselves distributed across a set of agents. There a good overall case may not be the one derived from the summation of best sub-cases.

A major difficulty is that the different knowledge sources give rise to wrong conclusions because of the ambiguity of terminology and the different goals for which the cases have been developed. In one source two terms can have different meanings and in the other different terms can mean the same.

One distinguishes different kinds of heterogeneity:

- (1) Heterogeneity with respect to the participating systems and the internal organisation.
- (2) Heterogeneity in the runtime environment, for instance, in hardware or operating systems.
- (3) Heterogeneity in the semantics. This results from the autonomous development of the systems; it occurs in many knowledge-based systems.

Here we will examine the third point only. A first step in handling this is the introduction of standards. In many disciplines, standards exist. Unfortunately, sometimes there are different standards that are not compatible with each other. Examples are the classification systems SNOMED and ICD in medicine.

A next step in this direction is the creation of ontologies. This raises the question of how to deal with heterogeneous ontologies. There are some limitations for this,

mainly because it may need too much effort. This is a major problem in database technology too.

12.6 Tools

Recio-García et al. (2005) describes how to make use of ontologies for CBR with jColibri.

Multiple groups studying ontologies have developed tools for ontology development. Some of them are:

- WebOnto, Knowledge Media Institute, Open University; see <http://projects.kmi.open.ac.uk/webonto/>
- Protégé, Stanford Center for Biomedical Informatics Research, Stanford University; see <http://smi-web.stanford.edu/projects/prot-nt>
- Ontosaurus, Intelligent Systems Division, Information Sciences Institute, University of Southern California; see <http://www.isi.edu/isd/ontosaurus.html>

12.7 Chapter Summary

This chapter started with a refined view on properties of cases as well as of case bases. Besides general properties, provenance and distributed case bases have been considered.

Next, the relation between similarity and utility was discussed in an informal way from the viewpoint of practical applications. Then the context concept was made precise. The types of contexts were structured from very general contexts on the top level down to more specific contexts on the lower levels. The relations to contexts were brought into a relation to the knowledge containers; this will later play a major role in developing systems. A quick look was given to distributed case bases.

The considerations in the generalized approach were deepened and investigated from the e-commerce viewpoint. The gap between functionalities and products was considered.

12.8 Background Information

The properties of cases can be viewed from many points of view. The investigations of these properties are scattered around the literature.

Both concepts of knowledge containers and adaptation-guided retrieval were introduced at the same conference, respectively by Richter (1995) and Smyth and Keane (1995).

The properties of case bases are studied in MacDonald et al. (2008). Case provenance is described in Leake and Whitehead (2007).

Distributed CBR has been reviewed in Plaza and McGinty (2005). Other works that have explored potential benefits from distributed case bases include Leake and Sooriamurthi (2001, 2003). Learning in distributed case bases is the topic of Prasad (2000). An application of distributed CBR for the Web can be found in Watson and Gardingen (1999).

Ontologies are explicit specifications of shared conceptualizations (Gruber 1995). An early development of an ontology for CBR is described in Díaz-Agudo and González-Calero (2000). An application for it is given by Recio-García et al. (2005). A system using ontologies for distributed CBR is given in Bichindaritz (2004). For distributed case bases and ontologies, see also Bouzeghoub and Lecocq (2005).

The importance of contexts is still growing, not only in CBR but in all of software development. Context awareness is reflected in many workshops on this topic. An early reference is Abowd et al. (1999). Becerra-Fernandez et al. (2006) discuss context-aware systems for knowledge management and include the role of CBR.

12.9 Exercises

Exercise 1 Construct an ontology for your favourite area of music that can be used by an e-commerce shop.

Exercise 2 Define two task ontologies for the same type of car, one for fault diagnosis and one for sale. Compare these two.

Exercise 3 Suppose you have to build a CBR system for supporting companies when they make decisions with respect to ecology.

- (a) What sources would you consider?
- (b) Which aspects for provenance would you consider?
- (c) Discuss in particular the temporal aspects!

Exercise 4 In the first semester one learns that the limit of the sum of two real-valued sequences is the sum of their limits. As homework one gets the task of proving the analogue statement for products. Make this analogy formally precise.

References

- Abowd GD, Dey AK, Brown PJ et al (1999) Towards a better understanding of context and context-awareness. In: HUC'99: handheld and ubiquitous computing. First international symposium, Karlsruhe, Germany, 27–29 September 1999. Lecture notes in computer science, vol 1707. Springer, Berlin, p 304
- Becerra-Fernandez I, Cousins KC, Weber RO (2006) Nomadic knowledge management systems: applications, challenges, and research problems. *Int J Mob Learn Organ* 1(2):103–121

- Bichindaritz I (2004) Mémoire: case-based reasoning meets the semantic web in biology and medicine. In: Funk P, González-Calero PA (eds) ECCBR 2004: advances in case-based reasoning. 7th European conference, Madrid, Spain, August/September 2004. Lecture notes in computer science (lecture notes in artificial intelligence), vol 3155. Springer, Berlin, p 47
- Bouzeghoub A, Lecocq C (2005) Use cases of heterogeneous learning ontologies. In: ICALT 2005: fifth IEEE international conference on advanced learning technologies, 2005. ICALT 2005. IEEE, Los Alamitos, p 881
- Díaz-Agudo B, González-Calero PA (2000) An architecture for knowledge intensive CBR systems. In: Blanzieri E, Portinale L (eds) ECCBR 2000: advances in case-based reasoning. 5th European workshop, Trento, Italy, September 2000. Lecture notes in computer science (lecture notes in artificial intelligence), vol 1898. Springer, Berlin, p 37
- Gruber T (1995) Toward principles for the design of ontologies used for knowledge sharing. *Int J Hum-Comput Stud* 43:907–928
- Leake DB, Sooriamurthi R (2001) When two case bases are better than one: exploiting multiple case bases. In: Aha DW, Watson ID (eds) ICCBR 2001: case-based reasoning research and development. 4th international conference on case-based reasoning, Vancouver, BC, Canada, July/August 2001. Lecture notes in computer science (lecture notes in artificial intelligence), vol 2080. Springer, Berlin, p 321
- Leake DB, Sooriamurthi R (2003) Dispatching cases versus merging case-bases: when MCBR matters. In: Russell I, Haller SM (eds) FLAIRS-2003: sixteenth international Florida artificial intelligence research society conference. AAAI Press, Menlo Park, p 129
- Leake DB, Whitehead M (2007) Case provenance: the value of remembering case sources. In: Weber RO, Richter MM (eds) ICCBR 2007: case-based reasoning research and development. 7th international conference on case-based reasoning, Belfast, UK, August 2007. Lecture notes in computer science (lecture notes in artificial intelligence), vol 4626. Springer, Berlin, p 194
- MacDonald C, Weber RO, Richter MM (2008) Case base properties: a first step. In: ECCBR 2008 workshop program. Workshop on uncertainty, similarity and knowledge discovery, Trier, Germany, 1–4 September 2008
- Plaza E, McGinty L (2005) Distributed case-based reasoning. *Knowl Eng Rev* 20(3):261–265
- Prasad MV (2000) Distributed case-based learning. In: Fourth IEEE international conference on multi-agent systems. Boston, MA, 10–12 July 2000. IEEE, Los Alamitos, p 222
- Recio-García JA, Díaz-Agudo B, Marco A, Wiratunga N (2005) Extending jColibri for textual CBR. In: Muñoz-Avila H, Ricci F (eds) ICCBR 2005: case-based reasoning research and development. 6th international conference on case-based reasoning, Chicago, IL, USA, August 2005. Lecture notes in artificial intelligence, vol 3620. Springer, Berlin, p 421
- Richter MM (1995) The knowledge contained in similarity measures. In: Keynote at ICCBR-95: 1st international conference on case-based reasoning, Sesimbra, Portugal, October 1995
- Smyth B, Keane MT (1995) Experiments on adaptation-guided retrieval in case-based design. In: Veloso MM, Aamodt A (eds) ICCBR-95: case-based reasoning research and development. First international conference on case-based reasoning, Sesimbra, Portugal, October 1995. Lecture notes in computer science (lecture notes in artificial intelligence), vol 1010. Springer, Berlin, p 313
- Watson ID, Gardingen D (1999) A distributed case-based reasoning application for engineering sales support. In: IJCAI-99: 16th international joint conference on artificial intelligence, Stockholm, Sweden, 1999, vol 1. Morgan Kaufmann, San Francisco, p 600

Chapter 13

Advanced Similarity Topics

13.1 About This Chapter

This advanced chapter is addressed to readers who are interested in foundational aspects, formal aspects, subjectivity, functional dependency, and other special additional problems. The general aspects of similarities are extended and deepened. First, a formal semantics for similarity measures is introduced that allows defining correctness for similarity-based computation. Furthermore, various problems and properties such as missing values, explanations, and so on, are discussed. Familiarity with Part I and Chaps. 6 and 7 on similarity is recommended.

13.1.1 Foundations

In Chap. 6, Basic Similarity Topics, the semantics, i.e., the meaning of similarity measures, was related to utility in an informal way. However, every serious scientific branch needs a formal foundation. We will now explore this. With respect to a similarity measure, one may ask questions like:

- Q1: What does $\text{sim}(a, b) = 0.85$ tell me?
- Q2: What does $\text{sim}(a, b) > \text{sim}(a, c)$ mean?
- Q3: Why should I be interested in the nearest neighbour rather than in the fifth nearest neighbour?

These questions are crucial for CBR systems and related to the semantics issue.

The semantics looks at functional properties of the measure. A similarity is, however, a computational device too. This raises the question of looking also at nonfunctional properties. First, we identify the properties of interest and we deal with two aspects:

1. Functional aspects: What does the similarity measure compute?
2. Nonfunctional aspects: How efficient is the computation?

The second aspect plays a role if there are several similarity measures where the similarity relation is almost the same but the computation time differs.

As mentioned in Chap. 2, Basic CBR Elements, two persons may use different measures for retrieving answers from a case base. The reason was that each of them interprets “more similar” as “preferable”. Therefore, different preferences lead to different measures. In other words, the intention of similarity is to describe utility. We will now investigate this more closely.

13.1.2 Formal Aspects

Here we assume the most general setting, where sim is defined on $U \times V$; U contains problems and V contains decisions or actions.

The definition of the semantics is analogous to the study of formal properties of programming languages. In programming languages, one has a specification and the task is that the specification be met. In CBR, one replaces the specification by the utility. Hence, one specifies a utility and the task is that the CBR system present the most useful available solution. In many situations, a formal specification is not exactly presented for program development, and in the same way a formal utility may not be available. In both situations, one has to be satisfied with approximating the intended goal as closely as possible.

13.1.3 Meaning and Semantics

First, we deal with the functional aspects. In this case, $\text{sim}(u, v)$ can be directly interpreted as the utility of v for u . Utility is based on preference relations and they have to be discussed first.

Definition 13.1

- (i) A preference relation is a partial ordering of the form $\text{pref}(P, S_1, S_2)$, read as “ S_1 is preferred over S_2 for problem P ” (or is “more useful than”).
- (ii) A utility function u is a real-valued function of the form $u(P, S)$, verbalized as “ $x = u(P, S)$ is the utility of S for P ”.

Although utility has its roots in economics, it is by no means restricted to monetary values. For instance, one prefers something that is more comfortable, closer, better looking, and so on. The reader has to notice that a preference relation represents a view. This does not mean that the preferred object, for instance, a preferred decision, is more useful to other users. In addition, a highly estimated utility does not guarantee success. This is investigated in the revise step of the CBR process model.

Some aspects of utility can be formulated mathematically, others not. We will discuss this in Chap. 22, Basic Formal Definitions and Methods.

The demand on a similarity measure is that it reflect or at least approximate the utility $u(Q, S)$ of the solution S for the user query Q . This means that for a problem P and a case (P, S) , the equation

$$\text{sim}(Q, P) = u(Q, S)$$

should at least be approximately true. In this view, the knowledge contained in a measure is concerned with the knowledge about the underlying utility function, which contains utility knowledge. In particular, the nearest neighbour to a query problem should give the most useful solution.

In addition, a measure can contain also efficiency knowledge. It may happen that the utility can be sufficiently well approximated by two measures, which may be different in the computational effort.

The preference relation on some set V of possible choices was defined as a partial ordering \geq ; it expresses what one likes more. Utility functions quantify this relation; they are real-valued functions

$$u : V \rightarrow \mathfrak{R}.$$

Now we look at the relation \geq_a induced by a measure sim :

$$b \geq_q c \Leftrightarrow \text{sim}(q, b) \geq \text{sim}(q, c)$$

and observe that \geq_q is a preference relation. One way to express this in natural language is “ b is better than c with respect to $q \Leftrightarrow b >_q c$ ”.

An obvious difference between the two concepts is that utilities have one argument and similarity measures have two arguments. The following concepts bridge this gap.

Each similarity measure sim and each problem $q \in U$ induce a utility function

$$u_{\text{sim},q}(b) := \text{sim}(q, b).$$

$u_{\text{sim},q}(\cdot)$ is defined on the elements of V .

If we consider utilities as primary, then similarity is a means to maximize utility. This leads to the view of taking utility functions as the format for specifications of a measure. More concretely, the utility functions play the role of the specification for the nearest neighbour(s).

One way to formulate the correctness of programming languages is by using the definition that the operational semantics computes the fixed point specification. In CBR, fixed-point search is hidden in the nearest neighbour search: One takes a case, tries to find a better one and iterates this. A nearest neighbour is then a fixed point.

Suppose that u describes the specification. For similarity systems the correctness proof then has to ensure

$$u_{\text{sim},q}(\cdot) = u(\cdot) \quad \text{for each } q \in U.$$

More precisely, one distinguishes three forms of correctness:

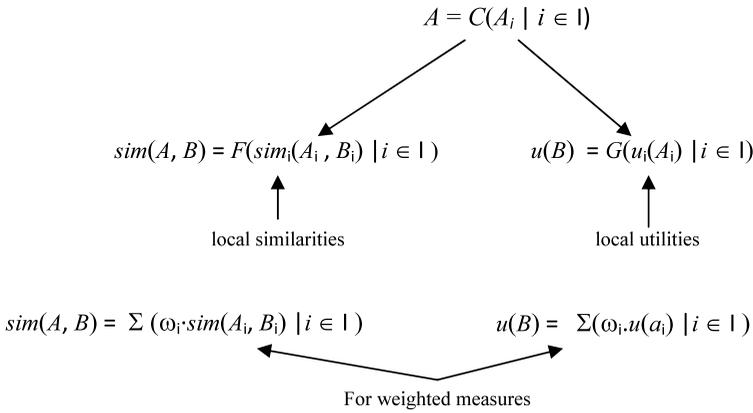


Fig. 13.1 The local-global principle for defining similarity measures

Definition 13.2 A measure *sim* and a nearest neighbour search is with respect to a specification:

- (1) Nearest neighbour correct, if the nearest neighbour for each query object has always the highest utility;
- (2) Relational correct if the preference relation of the $u_{sim,q}(\cdot)$ coincides with the preference relations of the $u_q(\cdot)$;
- (3) Totally correct if $u_{sim,q}(a) = u_q(a)$ for all q and a .

This allows us to present answers to the questions Q1, Q2, and Q3:

- Q1: $sim(a, b) = 0.85$ says that the utility of choosing b for a given a is 0.85 in terms of utility units.
 Q2: $sim(a, b) > sim(a, c)$ says that the utility of b is larger than the one of c for the problem a .
 Q3: The nearest neighbour is the most useful one.

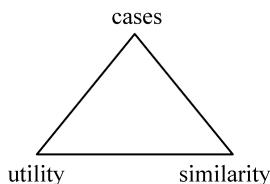
The last condition follows the *Maximum Expected Utility* principle (i.e., in case there is a probability distribution on the utilities).

The question remains about how one can define the similarity measure when the utility is given. This is part of the task of similarity assessment. A way to do this is to employ the local-global principle as shown in Fig. 13.1. In the first step, this description is applied for the utility and then the measure mimics this description. This should be done in such a way that:

- (1) Local similarities correspond to local utilities.
- (2) Global similarities correspond to utility functions.

This view assumes that both object description and utility use the same language. Unfortunately, often this is not the case, as was discussed in Chap. 3, Extended View, where we considered an example dealing with electronic switches.

Fig. 13.2 Compatibility triangle



More examples occur in the following situations:

- (a) $U =$ queries, $V =$ answers. For instance, the query can ask for a property and the answer can describe the property.
- (b) $U =$ demanded products, $V =$ offered products. The demand can consist of a description of the functionality (I want a car that uses little gasoline) while the product description has, for instance, the number of cylinders.
- (c) We want a design for smooth traffic. That is not even well defined and we return to it below when we discuss subjectivity. In any case, one can just describe the number of traffic lights and the speed limit, etc., but by no means how smooth the traffic is.

Therefore, one can weaken the condition $u_{\text{sim},a}(\cdot) = u(\cdot)$ by the requirement that $u_{\text{sim},a}(\cdot)$ and $u(\cdot)$ are “going in the same direction”.

For this the notion *concordant* is useful; it is defined as a relation between functions on a totally ordered domain \mathfrak{R} :

Definition 13.3 Two functions $f, g : X \rightarrow \mathfrak{R}$ are concordant if

$$D(f, g, x, y) := (f(x) - f(y)) \cdot (g(x) - g(y)) \geq 0 \quad \text{for all } x, y \in X.$$

A stronger version is when a strict inequality is required. This means both functions have the same monotonicity character, i.e., if one increases then the other one increases too.

Even if the utility is precisely defined, this does not mean that one can easily prove or disprove that a solution has a desired property. The reason is that the relation between the solution and the problem, i.e., the desired properties of the solution, is too complex or even unknown. Often, one can judge the properties of a solution only after using it, i.e., a posteriori. The only way out is to use the solution in reality or in a simulation. This is the revise step in the CBR process model and it is discussed in Chap. 10, Evaluation, Revision, and Learning. It is also the topic of learning procedures in that chapter.

As a consequence, in general one can expect only that similarity approximates utility. The introduction of utilities has the consequence that the compatibility of cases and similarities has to be extended to a compatibility triangle as in Fig. 13.2.

13.1.4 Subjectivity

Utility is not a context-independent notion like the concepts in mathematics or physics. It depends on the user and on the situation. One agent may dislike what

may be useful to another agent. This means, *utility is subjective*. If we base similarity on utility, similarity is subjective too.

The same applies to probabilities. In fact, there are great deficiencies in the classical model-based approach and they have been questioned in various ways.

One has to observe that *subjective* should not be confused with *irrational*. Subjective utility simply means that different persons in different situations may have different utilities and these may very well be precisely defined.

On the other hand, a subjective utility or probability can also describe an individual's personal judgment that is not precisely defined, for instance, how likely or how useful a particular event is. This is not based on any precise computation, but is often a reasonable assessment by a knowledgeable person. The problem of subjectivity is increased by the phenomenon that people cannot even describe precisely what they like.

As a consequence, there is the problem that utilities often are not precisely formulated or even known. That is familiar from software development: Initially, one rarely has a precise formal specification. Often, one cannot precisely formulate a specification either because it is not fully clear or because it is not fully known. It is the purpose of requirements engineering to collect as much information about the specification as possible. Another way is through agile programming, where requirements collection and programming are interleaved.

An obvious question can now be answered: Why do we need similarity at all if it should only mimic utility? The reason is that the solutions are not defined in terms of utility directly and similarity is eventually reduced to utility. The object description used by similarity does not contain what one wants to do with it.

An object is, for instance, a car, and a method could be the description of a motor. On the other hand, there are in general infinitely many possible utilities that one can associate with an object. As mentioned in Chap. 2, Basic CBR Elements, one can consider cars from the viewpoint of racing, driving the family, or selling them. All these views have different utilities. In such cases, it is almost impossible to associate the utility directly with the cars as manufactured objects because it is problematic to associate the value with the product description. For instance, how do we associate the property of being *comfortable* with a car description?

The subjectivity of utilities is a major reason for the existence of so many different similarity measures. Utility is connected with a goal and the goals are context-dependant. For instance, different companies can connect different goals with same object.

Therefore, we distinguish four different situations:

1. The utility is precisely defined and known. This is the simplest situation.
2. The utility is precisely defined but it is not (completely) known. This occurs in business everyday; companies do not want to make it public.
3. The utility changes dynamically because of a changing context.
4. The utility is informally known.

The last point creates many difficulties.

Examples are:

- Personal taste:
 - What is nice?
- Taste of a group:
 - What do customers prefer?
 - What kind of residential areas do citizens prefer?
 - Which TV shows do certain people like?
- Commercial interests:
 - What kind of investment do people prefer?
- Professional judgment:
 - Under what conditions does a medical image represent a pathology?

One can measure some preferences by observations, for instance, customer preferences. In the case of professional judgment, groups often agree in their judgment. However, this does not mean that they are able to give a definition.

These topics influence recommender systems. The detection of a similarity measure is important for recommendations that ideally should match the taste of the customer. We discussed this in Chap. 3, Extended View.

The question is also of interest in recognising visual information. In cognitive science, one has paid much attention to the problem about the conditions under which humans decide that two objects “are similar”. This runs under the heading *same-different* problem. The research in this area has concentrated on looking at different features as in the Tversky contrast model. Many applications are in medicine and image retrieval.

Although humans are in general not able to quantify their utility, they are usually able to provide preference relations in individual instances. They do this without being able to justify it completely. From a preference relation one can generate a similarity relation.

If humans know their preferences in certain instances, one can observe their behaviour. One can also ask questions to make this information more precise. In other words, one can collect cases.

In such situations, Machine Learning plays a role. Given the observations, one starts a learning process with the goal of getting a similarity (or utility) concept for a person or a group of persons in a certain context. We discuss this in Chap. 10, Evaluation, Revision, and Learning.

In this context, the local-global principle is of use. One can proceed in two steps, parallel to the procedure where formal utilities exist:

1. Represent the object structure for the utilities informally;
2. Replace the informal entries by precise estimates, default values, and so on.

Mixed problems occur if the constructor function has to combine two or more local utilities of heterogeneous nature. This is, in particular, difficult if one has to integrate model-based and subjective components. Often, the data structures between

the two are not even compatible. To make things worse, utilities may compete and change dynamically. When combining subjective preferences and similarities, the constructor function has also a subjective character.

Utility problems occur in wicked problems. Although wicked problems do not have a precise definition, they have several characteristics. Some major ones are:

- The problems are of mixed nature and contain model-based, subjective, and context-dependent elements. One has different participants with different preferences and this makes it problematic to judge the quality of the solution.
- The influence factors for the utility and hence for the similarity measure are difficult to determine and one can often judge the quality of a solution only a posteriori.
- The participants may change their preferences dynamically.

The combination of different subjective utilities has in addition the problem that it is not clear what the aggregated utility should look like. Here, cases are often useful. They can tell us when a solution to a problem was well accepted. In such a situation, one can use the acceptance rate to measure the utility.

13.1.5 Discussion of the Axioms for Similarity

Due to the changed view on similarity and measures in the last few years, axioms for similarity have been more or less given up at least in general. Originally the measure was concerned with the similarity of problem situations in order to apply previous experiences in some kind of analogical reasoning. The first idea of similarity between objects was that *they looked similar* and this property is clearly symmetric and reflexive. In this view similarity was thought of as a form of fuzzy equality. Most axioms discussed earlier are still in use but not in general. From a naïve point of view it seems intuitive to assume axioms like reflexivity and symmetry. In the past, one questioned the axioms only occasionally in cognitive science in connection with the Tversky measure.

- The reflexivity axiom makes no sense if similarity describes partnership, especially when the partners come from different sets. For instance, partners can be products and functionalities.
- A simple example where the symmetry axiom fails is that of distances in a city because of one-way streets. It can also be considered from the point of view that we deal with questions and answers. If an experience case is similar to an actual query then one cannot automatically deduce the inverse similarity. In general, symmetry can fail in the context of utility. For example, the degree of usefulness of A for B may not be the same as the degree of usefulness of B for A .
- The transitivity axiom of equality was abandoned because small errors add up.
- For similarity relations another standard argument against transitivity is when one considers:

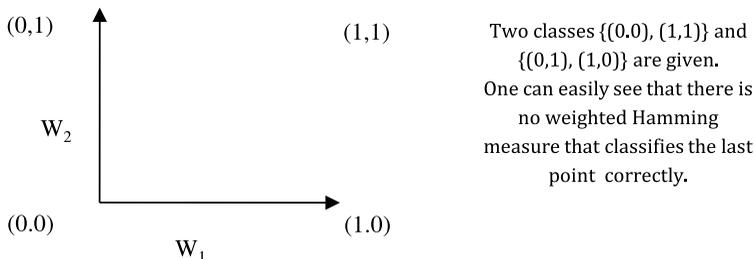


Fig. 13.3 Violation of monotonicity

- (a) Mr. Sarkozy and Mr. Obama are similar because both are presidents.
- (b) My neighbour Francois and Mr. Sarkozy are similar because both are French.
- (c) But Francois is not similar to Mr. Obama.

This argument is, however, not really convincing. What happens here is that two similarity concepts are mixed. Therefore, it is no surprise that some confusion arises. Often, one sees a basic error here by assuming that there is a universal similarity measure. As we saw, there are many similarity concepts, depending on the intended utility, and here one is mixing two similarity predicates. A lesson is that even in informal conversations one should always use the phrase “*A* and *B* are similar with respect to . . .”.

Hence we can ask: Are we running out of axioms and will a similarity measure be just an arbitrary real-valued binary function? Besides the fact that the axioms often hold, there is a hidden but very useful additional axiom. It is connected with the local-global principle for similarity measures. We consider a global measure sim and local measures sim_i .

The Monotonicity Axiom:

If $\text{sim}(a, b) > \text{sim}(a, c)$, then there is at least one $i \in I$ such that $\text{sim}_i(a_i, b_i) > \text{sim}_i(a_i, c_i)$.

This axiom can be regarded as a partial order form of the substitution axiom (substituting equals by equals) for equalities. The importance of the axiom is twofold: If it holds, it allows in general more efficient computations and it simplifies the assessment of utility functions and similarity measures. The point is that local improvements of measures also lead to a global improvement. One can make use of this in learning similarity measures.

Many measures previously introduced satisfy this axiom. The question arises about whether the axiom can always be satisfied by a suitable choice of the measure. An example where it fails is the well-known XOR classification problem that is described in Fig. 13.3.

Suppose there are two binary attributes, which gives four cases, and two classes. Assume the case base already contains three elements that are correctly classified. Then there is no weighted Hamming measure for classifying the fourth element correctly using the nearest neighbour method:

If the monotonicity axiom fails then there must be some hidden relation between the cases that is not directly expressed by the attributes. The question is, how can the measure be improved when dealing with such situations? The presented XOR example shows that there are two choices only:

- Take a very complicated and not intuitive measure;
- Extend the vocabulary.

The second possibility provides a way out. It is the introduction of additional *virtual* (definable) attributes in addition to the original primary ones, as discussed before. In case of the XOR problem, the attribute $\text{XOR}(x, y)$ will suffice. The virtual attributes shift nonlinear dependencies between attributes from the measure into the definition of virtual attributes. There is a relation to neural nets where the XOR example shows that with a single neuron only, one cannot compute all Boolean functions.

If one looks at CBR from an optimization point of view, one wants to maximize $\text{sim}(P, S)$ for a given P . For monotonic measures this is a classical multi-criteria optimization problem that we will not discuss here.

13.1.6 First- and Second-Order Similarities

For cases (problem, solution), we have studied utilities and similarities so far. Now we call these first-order utilities and similarities.

In practical situations, however, it occurs frequently that we have different agents who offer us different utilities and similarities. That is because both are not precisely known or formulated and the agents have different knowledge. This raises the questions:

- How different are they?
- Which ones should I choose?

If one makes a wrong choice, in the lucky situation it could be tolerated, but this could also lead to unwanted errors. For finding out the kinds of errors, one has to compare utilities and similarities. We call this a second-order problem. For comparing utilities and similarities, we again use similarity measures, but now on a higher level.

Suppose we have two utilities u_1 and u_2 and measures sim_1 and sim_2 on a domain U . We consider the presentation of objects from U to the utility functions as random events. This allows defining similarity measures sim_u and sim_{sim} . Here E denotes the expected value.

Definition 13.4

- (i) $\text{sim}_u(u_1, u_2) = E(|u_1(x) - u_2(x)| \mid x \in U)$.
- (ii) $\text{sim}_{\text{sim}}(\text{sim}_1, \text{sim}_2) = E(|\text{sim}_1(x) - \text{sim}_2(x)| \mid x \in U)$.

This says that the similarity of utilities and the similarity of similarity measures are measured by the average difference they provide. The second-order similarity compares the differences from a statistical point of view, which reflects the view that utility theory is mostly of statistical character.

The estimates may be more or less reliable but the comparison can provide a measure for risks associated with wrong choices. This plays a role when one of the similarity measures is more efficient to compute and one wants to know what can happen if one takes it.

In many situations, one has linear weighted measures. Frequently, the situation is that two measures differ by their weights and one asks which weights to choose. The simplest measures are those where all weights are equal. We call this weight vector ω_{triv} . The question is, Can we tolerate neglecting the weights? This would simplify the situation to a large degree. For this purpose one has to compare the given weight vector with ω_{triv} . In the ideal case, one would like to have a numerical value defined on the weight vectors to decide this. For this purpose, we introduce the notions of weight average and weight diversity. Suppose there are n attributes given with corresponding weights $\omega = \{\omega_i, 1 \leq i \leq k\}$.

Definition 13.5 The average weight ω_{av} of ω is

$$\omega_{\text{av}} = \frac{1}{n} \cdot \sum_{i=1}^k \omega_i.$$

If the weights differ only a little then the error obtained by omitting them at all can be negligible. This is, in particular, the case when the weights are just superficial estimates, which is quite often the case.

Definition 13.6 The weight diversity of ω is

$$\text{diversity}(\omega) = \frac{1}{k(k-1)/2} \sum_{i=1}^{k-1} \sum_{j=i+1}^k |\omega_i - \omega_j|.$$

For the trivial weight vector, diversity (ω_{triv}) = 0 holds. One can regard the diversity as an estimate of the distance between the measure with weight vector ω and the measure ω_{triv} .

For judging the weight diversity, we can introduce a threshold θ and use it as follows:

If $\text{diversity}(\omega) < \theta$ then make all weights equal; otherwise leave them as they are.

The choice of θ depends on the tolerated error. If the diversity is small enough for a whole class of problems then the trivial weights will suffice.

These considerations can be used in two ways:

- (1) Determine if two agents provide significantly different measures. If so, the user has to search for additional information. In particular, attributes of high weight where the agents differ can provide useful explanations.

- (2) For weighted measures, one can try to simplify the situation by neglecting the weights.

13.2 Miscellaneous Topics

In this section we discuss additional questions that arise occasionally. They should be consulted whenever they become pertinent.

13.2.1 Nonfunctional Aspects

These aspects do not deal with the result of the similarity concept itself but rather with the computation process. Of course, there are more nonfunctional elements coming from human-computer interaction, but we do not discuss them here. In Chap. 11, Development and Maintenance, we sketched them under properties of the periphery.

As aforementioned mentioned, the efficiency of the similarity computation is very important. We denote the time to compute $\text{sim}(x, y)$ as $\text{comptime}_{\text{sim}}(x, y)$.

Suppose now a probability distribution $\text{Prob}(x, y)$ for the occurrence of two objects x and y is given. Then we define the computation complexity as an average.

Definition 13.7 The computation complexity of sim is

$$\text{complex}(\text{sim}) = E(\text{comptime}_{\text{sim}}(x, y)).$$

Here E is the expected value.

A major aspect of the complexity of the representation language is that complex representations require more effort in the similarity computation:

- Object-oriented representations need costly computations.
- Flat attribute representations need less effort. For instance, Hamming or Euclidean measures can be computed fast.

This calls for a decision in case one has the choice of different representations. Although complex representations are more accurate, the question is whether they result in better solutions. For instance, the inputs may be of little exactness, so that a precise similarity computation is obscure anyway.

13.2.2 Top-Down Versus Bottom-Up

The local-global principle gives rise to two tasks for measures (see Chap. 2, Basic CBR Elements):

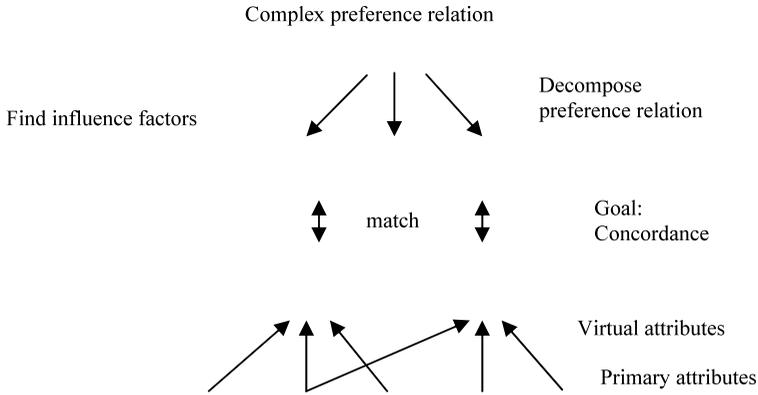


Fig. 13.4 Composition in synthesis tasks

- (a) The decomposition task: Break the object or concept down into atomic parts. This task often occurs because the object may be presented globally and the parts are initially unknown.
- (b) The synthesis task: Composes an object or concept from simpler parts. This task occurs in planning, construction, or configuration. This is shown in Fig. 13.4.

A preference relation and a utility are usually given in a holistic way: One knows what one wants and what one does not want. For computational reasons one has to break this down into individual parts. On the other hand, for defining a similarity measure, it goes in the opposite direction: One has only the primary attributes of the objects at hand. These are often not important by themselves; what matters is their combination. This leads to a synthesis procedure resulting in virtual attributes.

In general, both tasks play a role in relating the concepts of interest. For the objects under investigation, the local-global principle has very different realisations. The point is the unified use of the principle in order to allow a systematic treatment of the different techniques. For this purpose we will briefly introduce these techniques and discuss them from the point of view of unified use of the principle.

The construction process defines a kind of hierarchy where primitive local measures are the basic elements. This, however, does not prescribe a systematic bottom-up or top-down way for constructing the measure.

Bottom-up constructions follow the rules:

1. Define the local measures.
2. Define the relevance of the local measures.
3. Define an aggregation function.

This is a product-oriented view and involves a procedure. In many situation one builds up products in this way and it is natural to construct similarity measures in this way.

The top-down construction is oriented to the overall utility of the intended product and the similarity measure wants to match this. In order to obtain more infor-

mation about this utility it is decomposed top-down and the similarity definition follows this approach too. Figure 13.4 shows also the use of virtual attributes which are somehow intermediate between the top-down and bottom-up approaches.

Therefore, the construction of the measure and the utility function can go in opposite directions: The measure is often defined bottom-up while the utility has more a top-down decompositional character. The user provides the utility as a whole and it has to be analysed while the system designer starts with the attributes first and ends up with the global measure.

13.2.3 *Jumps and Noise*

Local measures should contain knowledge coming from the distribution of the values of the attributes. This means that local measures can and should contain specific information about the attributes.

We refer to the example of water temperature introduced before. Here we have the attribute WT with the domain, for example, $\text{dom}(\text{WT}) = [-100, +200]$. In numerical domains, we can separate regions in which the similarity undergoes little changes by using landmarks that separate the real axis into intervals. In each interval, water shows the same behaviour. For water temperature we identify three regions: $[-100, 0]$, $[0, 100]$ and $[100, 200]$. With respect to these regions, the measure allows some kind of qualitative reasoning.

Looking at the point 0 on the real axis, one observes that the similarity function makes a jump. This means that temperature values with arbitrary small numerical differences can lead to very different similarity measure values and therefore to completely different solutions. If these values result from physical observations, small variances are unavoidable and noise can lead to errors. Hence, one cannot trust the similarity computation in such situations. In terms of rough sets, there is a “do not know” area around the point 0.

The noise can occur in the problem as well as in the solution. Noise can create uncertainty which is discussed given in Chap. 15, Uncertainty.

13.3 Functional Dependency, Unknown and Redundant Values

Often, the similarity computation is corrupted because of special properties of the attributes. We consider some of such properties.

13.3.1 *Functional Dependency*

Attribute A is called functionally dependent on the attributes A_1, \dots, A_n if the values of A can be computed by a function f on the values of A_1, \dots, A_n . The func-

tional dependency may be conditional, i.e., it may only hold if certain other attributes have specific values. This is a special case of general dependency.

Functionally dependent attributes require special attention. If two objects agree on the value of some attributes, they will also agree on any functional-dependent attribute, but this should not increase the similarity between the objects. Otherwise, one could increase the similarity indefinitely by introducing an unlimited number of dependent attributes. However, if attributes agree on a functionally dependent attribute, they do not necessarily agree on the base attributes. This may cause some problems, and the impact on utility has to be verified. It is therefore an important task of knowledge acquisition to detect functional dependencies for the similarity computation.

Besides functional dependency, there is a more general kind of dependency. It means that certain values or properties depend not only on just one attribute but on several attributes. This plays a role when we consider weighted linear measures, as they require general independence.

13.3.2 Unknown Values

If some attribute-value is missing, i.e., has the value *unknown* in a case *C*, the computation of the similarity to a query incorporates an additional uncertainty. For dealing with this problem, there are different possible ways.

If the value of an attribute in a case is unknown, then one has the following choices:

- default value;
- a value which contributes to the similarity;
 - a maximal value (optimistic view);
 - a minimal value (pessimistic view);
 - a value with the highest likelihood.

These are the main choices but many other approaches exist to treat these situations.

13.3.3 Redundant Values

A query can contain redundant attributes in the sense that a smaller subset of attributes expresses the same overall need. When product wishes are acquired interactively through a series of questions by the sales agent, redundancy should be avoided since answering redundant questions requires unnecessary effort by the customer. We will discuss this in Chap. 20, Conversational CBR.

One can handle the situation in two ways:

- Ignore redundant attributes if one can ensure that the cases from the base have only values necessary for the solution.

- Omit the case S from the base and replace it by the query case P if S and P have different solutions. Even if S had provided a correct solution in the past, it was only a good guess and the attribute-value in the past was different from the value in P .

Redundant values can also lead to a wrong choice of solutions.

13.4 Additional Problems

We consider two problems where similarity is of interest.

13.4.1 Similarity and Explanations

Often, in the context of nearest neighbour search some user questions arise:

- Why is this one the nearest neighbour? Or, why is its solution useful for me?
- Why is another case not the nearest neighbour?
- Why is this case eliminated?

These questions call for an explanation. We consider two cases C and D for which we want an explanation. If a weighted similarity measure is used, the measure itself gives an answer. The principle is to investigate the attributes A that have high weights. This means we neglect attributes of low weights.

Question (a): If C and D agree on A , this indicates why C and D should be similar.

Question (b): If C and D disagree on A , this indicates why C and D should be different.

Question (c): If the local measure in C is very small, this indicates why it is eliminated.

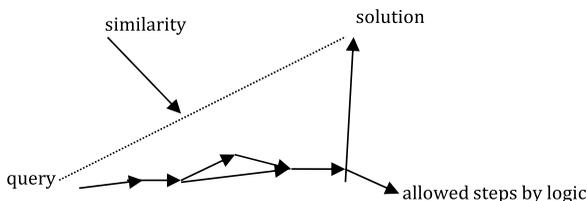
The first problem occurs if one wants to confirm a medical diagnosis. The second situation takes place if one is confronted with a decision which seems to be problematic and which one subjectively likes to reject.

Cases that contain the derivation of the results are themselves of some explanatory character. In general, a case representation does not need just the pair (problem, solution). It can contain arbitrary additional information, last but not least of explanatory character.

13.4.2 Similarity and Logical Inference

Logical inference is a type of reasoning that is very different from CBR reasoning. Similarity allows of drawing conclusions from previously stored cases in terms of

Fig. 13.5 Similarity and logic paths to an answer



how useful some object is. Logical inference does the same but in a very different way: One always wants to be on the “safe side”; correctness has the highest value. This means that logical reasoning leads from true statements to true conclusions; this is not the case for similarity reasoning. On the other hand, logical reasoning never discovers facts that are in principle not or only partially known. It can only make implicit facts explicit. Similarity reasoning is interested in useful conclusions that may not be known before.

A challenge is to combine similarity and logical reasoning: To be both, useful and correct. The problem is that both reasoning methods know little about each other. In particular, logic does not know if a conclusion or a set of rule applications leads to a higher similarity. One can compare this with a hill climbing problem where we see the top but in order to get there we have to follow certain paths. Doing so, we will never get lost but may not reach the top. This was discussed in Chap. 9, Adaptation.

We need to equip the logical rules with some knowledge about the landscape provided by the similarity measure. This knowledge then could guide the selection of the rules. The difference between similarity and logical reasoning is illustrated in Fig. 13.5, where the dotted line indicates the process from the query to the solution, we are providing an answer, the diagonal line is the line from the query and the answer.

The main areas where such situations occur are when CBR solutions need to be adapted. The adaptations are mostly defined by logical rules. In order to bridge the gap between logic and similarity, in general one needs additional domain knowledge that contains among other things:

- knowledge about when the application of a rule increases the similarity
- knowledge about which rules are likely to increase similarity.

One should keep in mind that the similarity is the primary one because it reflects utility while logical steps are simply allowed or forbidden. Again, we refer to Chap. 9, Adaptation.

13.5 The Knowledge Contained in the Measures

In Chap. 2, Basic CBR Elements, we presented the similarity measure in a knowledge container as an essential element for performing reasoning in CBR. The given examples show that measures can contain very different amounts of knowledge.

To discuss this we assume a fixed case base CB . In Chap. 10, Evaluation, Revision, and Learning, the notion that a measure sim_1 is *better informed* than the measure sim_2 was introduced. This is closely related to the information-theoretic view that has led to the definition of the relative importance of attributes. This leads also to the distinction between knowledge-rich and knowledge-poor measures. This is not a sharp classification; it is more or less a distinction.

As already indicated several times, there is a strong relation to the vocabulary container:

A similarity measure can only be as good as the vocabulary allows.

This was a major reason for introducing virtual attributes.

Knowledge-poor measures are mainly oriented towards syntactic properties of the representation. The advantage is that they can be applied universally, can easily be programmed and can allow efficient retrieval. The disadvantage is that they have no relation to domain knowledge and therefore do not allow a good approximation to the intended utility in more complex situations. They are extensively used in pattern recognition.

Examples of knowledge-poor measures are the Hamming measure and the Euclidean distance. Introducing weights adds relevance knowledge.

Knowledge-rich measures encode as much domain knowledge as possible. This means they are concerned not only with syntactical aspects but also with the semantics of the attributes. The advantage is a better approximation of the utility. The disadvantages are that they require high knowledge acquisition effort, they are very domain- and context-specific and often their retrieval is not very efficient. One finds many knowledge-rich measures among transformational, taxonomic and information content measures.

The knowledge is contained in both the local measures and the aggregation function F of the global measure. The numerical local measures shown in Chap. 6, Basic Similarity Topics, are already knowledge-intensive.

The aggregation function operates on the partial constructs. The important knowledge here is concerned with influences the parts have on the intended utility. High influence results in relevance that has to be respected by the measure. In linear measures this is the job of the weights.

For similarities that compare actions or behaviour, the fuzzy character (see Chap. 15, Uncertainty) of measure plays a dominant role.

Because utility can be concerned with many different aspects such as money, time, needs, and social environment, among others, a well-informed measure needs to contain knowledge about these areas. In addition, they should contain knowledge about the users.

Finally, we want to mention that a measure with much or little knowledge will not always result in bad or good solutions of the CBR system. This is because there is also the adaptation container that can improve solutions. It is the task of the system designer to decide about such design issues.

13.6 Tools

Most of the popular tools can deal with the problems touched upon in this chapter. The CBR tool myCBR (<http://mycbr-project.net>) offers not only solutions but also explanations.

13.7 Chapter Summary

The first part of this chapter discusses the concept of semantics, i.e., the meaning of similarity measure. The semantic is based on utility and allows formal specifications. As a consequence, one can talk about the correctness of similarity computations. This allows also for discussing several possible axiomatic properties. Often, the utility is unknown and/or not formally defined. For this purpose, subjectivity is investigated. For this reason the subjectivity of the utility of the context dependency is central. In summary, this leads to the result that one cannot expect total correctness of a similarity computation in general, but at most an approximate correctness.

In order to compare similarity measures from an abstract point of view, we study the knowledge contained in the measure. For constructing a measure, we compare a bottom-up and a top-down approach. The introduction of weight diversity helps in choosing weights.

The second part of this chapter discusses several aspects that can influence the use of similarity as noise, or as missing or redundant values. We show the problems and indicate possibilities to handle them. Besides finding a solution, the use of similarities relates to other tasks and methods. We discuss logical inferences and explanations.

13.8 Background Information

In programming languages, the concepts of semantics, specification and correctness play a central role. The same applies to any computational device and hence also to CBR systems, but this was not in the same way fully explored.

Subjectivity is also discussed in probability theory; see (Savage 1954). Subjectivity means that probabilities are not always based on mathematical models and estimates of data but sometimes rather on personal opinions. How to do this in a rational way is explained in Fishburn (1986). See also Künnapas and Künnapas (1974).

Utility theory is not restricted to financial aspects. It applies to any situation where one prefers one of two objects. The origin of mathematical utility goes back to Daniel Bernoulli (Bernoulli 1954), who also formulated the principle of diminishing utility as a concave function, which is standard today. Utility had been already discussed in ancient times. Cicero claimed that utility is subjective (Niegorski 1984). The modern approach by von Neumann and Morgenstern develops utility in a probabilistic context; see von Neumann and Morgenstern (1944). It relates utility closely to money; while we take a broader view here.

The purpose of the influence potential is to see how much a subset of values of the input pattern determines the class of the pattern. Subsets with low influence potential can be omitted; see Li (1993). This reduction simplifies the computation of the similarity measure. In CBR, it simplifies also the similarity assessment.

Wicked problems were first considered in urban planning; see Rittel and Webber (1984). Urban planning provides a rich source of wicked problems.

More on top-down versus bottom is in Stahl (2002). There, the topic jumps and noise is discussed under the view that similar problems may have dissimilar solutions. However, that refers to a naïve view of similarity.

A brief review of methods to deal with unknown values is given in Bogaerts and Leake (2004). Explanations for CBR solutions are discussed by Roth-Berghofer (2004). The relation between logic and similarity is discussed in detail in Mougouie (2009).

13.9 Exercises

Exercise 1

- (a) How would you define a poll for finding out preferences of used cars?
- (b) How would you define a similarity measure on this basis?

Exercise 2 Describe a sales situation with redundant query values and find out how to reduce them properly.

Exercise 3 Design a case base for a set of products and a set of potential buyers whose utilities are very different from each other.

Define a similarity measure for it.

Exercise 4 Describe a sales situation with incomplete query values and find out how to reduce them properly.

Describe two approaches to implementation.

Exercise 5 Describe a medical situation with missing values. How would you define a similarity measure?

References

- Bernoulli D (1954) Exposition of a new theory on the measurement of risk. *Econometrica* 22(1):23–36
- Bogaerts S, Leake DB (2004) Facilitating CBR for incompletely-described cases: distance metrics for partial problem descriptions. In: Funk P, González-Calero PA (eds) ECCBR 2004: advances in case-based reasoning. 7th European conference, Madrid, Spain, August/September 2004. Lecture notes in computer science (lecture notes in artificial intelligence), vol 3155. Springer, Berlin, p 62

- Fishburn PC (1986) The axioms of subjective probability. *Stat Sci* 1(3):335–345
- Künnapas T, Künnapas O (1974) On the mechanism of subjective similarity for unidimensional continua. *Am J Psychol* 87:215–222
- Li X (1993) Potential analysis for massively parallel computing and its application to neural networks. Dissertation, University of Kaiserslautern
- Mougouie B (2009) Integration of similarity-based and deductive reasoning for knowledge management. Dissertation, University of Trier
- Nicgorski W (1984) Cicero's paradoxes and his idea of utility. *Polit Theory* 12(4):557–578
- Rittel HWJ, Webber MM (1984) Planning problems are wicked problems. In: Cross N (ed) *Developments in design methodology*. Wiley, New York, pp 135–144
- Roth-Berghofer TR (2004) Explanations and case-based reasoning: foundational issues. In: Funk P, González-Calero PA (eds) *ECCBR 2004: advances in case-based reasoning*. 7th European conference, Madrid, Spain, August/September 2004. *Lecture notes in computer science (lecture notes in artificial intelligence)*, vol 3155. Springer, Berlin, p 389
- Savage JL (1954) *Foundations of statistics*. Wiley, New York
- Stahl A (2002) Defining similarity measures: top-down vs. bottom-up. In: Craw S, Preece A (eds) *ECCBR 2002: advances in case-based reasoning*. 6th European conference, Aberdeen, UK, September 2002. *Lecture notes in computer science (lecture notes in artificial intelligence)*, vol 2416. Springer, Berlin, p 406
- von Neumann J, Morgenstern O (1944) *Theory of games and economic behavior*. Princeton University Press, Princeton

Chapter 14

Advanced Retrieval

14.1 About This Chapter

The first part of this chapter is addressed to readers who deal with complex objects and need advanced retrieval algorithms. The methods are complex indexing methods. Also, fuzzy retrieval is discussed. The second part of the chapter deals with principal and advanced retrieval problems. A basic question is about how the search space can be reduced. This chapter requires knowledge from Chap. 5, Case Representations, Chap. 6, Basic Similarity Topics, Chap. 7, Complex Similarity, and Chap. 8, Retrieval.

14.2 General Aspects

The methods in this section deal with more or less complex situations and complex case structures. In these situations, one needs appropriate case representations. For this reason, the methods require special techniques for representing cases. These representations are not covered by the methods shown in Chap. 5, Case Representations, but extend them.

Retrieval requires adequate index structures, and the ones we cover here are of a complex character. We start with two advanced indexing and retrieval methods:

- Case retrieval nets are suited to large case bases;
- The fish and shrink method is suited to smaller case bases where the cases are highly complex.

14.2.1 Case Retrieval Nets

This method is of practical interest because it can handle large case bases. It interweaves case representation, similarity computation, and retrieval in an interesting

way and it is related to textual and conversational CBR. In Chap. 17, Textual CBR, we discuss this again. The purpose is to find an answer case to a query in an incremental way.

First, the case representation is somewhat special. The basic elements are:

- The global structure is an annotated net with input nodes, inner nodes, and output nodes.
- The basic idea is to decompose a case into elementary information entities IE, for instance
 - attribute-values
 - keywords
 - numerical values
 - word forms for texts

That means the representation is not only attribute-based. The information entities are annotations of the nodes of a net (a graph).

- Cases as well as queries are sets of information entities.
- At the input nodes one finds the problem input, and at the output nodes, the solutions. Because there are several input nodes, one can add inputs piecewise and complete the query systematically. This is analogous to the way done in Chap. 20, Conversational CBR. The incremental input is provided by the rules that are, however, applied automatically.
- Cases are represented as specific sets of information entities.
- Special nodes are case nodes that are annotated with cases.
- The annotations of the edges are the local similarities between node annotations.
- One can activate the nodes. Activations are denoted by real numbers and can be changed and propagated. Activation means that the nodes are participating in the actual process of finding a solution.
- Initially the nodes corresponding to information entities of the query are activated; the activations are propagated through a net according to the local similarities until they reach the case nodes.
- The activity at the case nodes represents the similarity to the query.

Next, formal definitions and algorithms for these ideas are given.

Definition 14.1 A Basic Case Retrieval Net (BCRN) is a 5-tuple $N = (E, C, \text{sim}, \text{rel}, \Pi)$ such that:

- I. E is a finite set of information entities IE.
- II. C is a finite set of case nodes and their annotations.
- III. sim is a similarity measure: $\text{sim} : E \times E \rightarrow \mathfrak{R}$; $\text{sim}(e, e')$ describes the local similarity between two information entities e and e' . Nodes with zero similarity are not connected.
- IV. rel is a function called relevance function; $\text{rel} : E \times C \rightarrow \mathfrak{R}$; $\text{rel}(e, c)$ describes the relevance (the weight) of the information entity e for the case c .
- V. Π is a set of propagation functions $\pi_n : \mathfrak{R}^E \rightarrow \mathfrak{R}$ for each node $n \in E \cup C$.

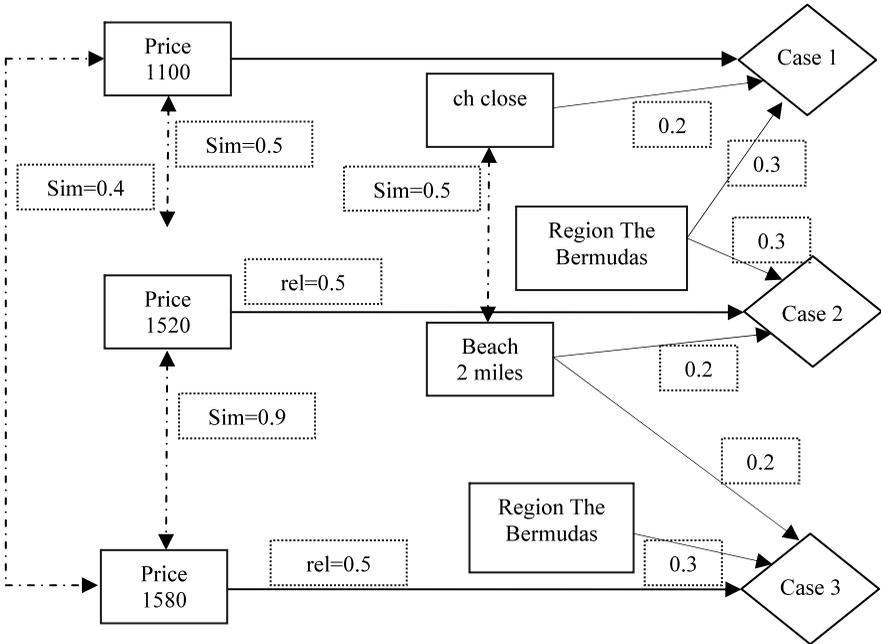


Fig. 14.1 Travel example

The generation algorithm for basic case retrieval nets follows:

```

FOR(all  $c \in CB$ )
  Insert a case node  $n(c)$  for  $c$  in BCRN
  FOR(all IEs in  $C$ )
    IF(BCRN does not yet contain the IE  $e$  THEN
      insert the IE  $e$  in the BCRN
    END IF
    insert a pointer from  $e$  to the case node  $n(c)$  (relevance edge)
  END FOR
END FOR
FOR all IEs of the case base
  insert bidirectional similarity edges to all IEs, that belong to this IE
  and compute the local similarities
END FOR
    
```

There are similarities between information entities and these entities have relevance to the cases as in the previous approaches. The propagation functions compute the global similarity.

Next, we illustrate this by a travel example in Fig. 14.1. The IEs have the instances of Price, Distance to the Beach, and Region. The dashed arrows are anno-

tated with the local similarities. The other arrows are annotated with the relevancies (the weights). There are three cases shown. Input units are not represented as well as the propagation of activations, and therefore a winner is not shown; we discuss this next.

Now we describe the activation of the nodes that start on the information entities of the query. The activation will generate the similarity of the query to the possible solution cases. This goes stepwise and can be compared with the summation steps in weighted linear measures.

Definition 14.2

- (1) An activation of a BCRN is a function $\alpha : E \cup C \rightarrow \mathfrak{R}$.
- (2) The activation of the query q is defined by

$$a_q(e) = \begin{cases} 1 & \text{for the nodes } e \text{ of the problem,} \\ 0 & \text{otherwise.} \end{cases}$$

This activation is propagated in discrete time steps t through the net.

Definition 14.3 The activation at time t is a function $\alpha_t : E \cup C \rightarrow \mathfrak{R}$ determined by:

- (1) An initial activation: Activation of the query, i.e., $\alpha_0(e) = 1$ if the IE e occurs in the query and 0 otherwise;
- (2) Propagation step: For an IE e that has a connection to an activated node:

$$\alpha_{\tau+1}(e) = \pi_e(\text{sim}(e', e)\alpha_t(e') | e' \in E).$$

The propagation takes care of the similarities and the relevancies. The activation of the case nodes reflects the similarity of the query node, which is computed by the propagation function.

The cases themselves are represented by a subgraph with one case node and all IE nodes with which it is connected.

The result is obtained when cases c are reached:

$$\alpha_{\tau+1}(c) = \pi_c(\text{rel}(e', c)\alpha_t(e') | e' \in E).$$

As mentioned, propagation steps correspond to the summation of the local similarities in a weighted sum for obtaining the global similarity and the activities correspond to the presently known similarity to the query.

The cases are partially ordered according to their activation, i.e., according to the present similarity to the query. The case with the largest activation will be the winner, i.e., the nearest neighbour to the query.

For the retrieval, first the activations of the nodes are initialized, the query IEs as 1 and all others as 0. Then two kinds of propagation steps take place, one for

the similarity and one for the relevance. In this way, the nearest neighbour to the query is determined.

In pseudocode, the retrieval algorithm reads as follows:

```

Input: Basic Case Retrieval Net  $(E, C, \text{sim}, \text{rel}, \Pi)$  and query  $q$ .
Output: Sorted list of cases.
Procedure Retrieve( $(E, C, \text{sim}, \text{rel}, \pi), q$ )
VAR  $\alpha$ : array  $[1 \dots |E \cup C|]$  of  $[0, 1]$ 
BEGIN
  FOR all  $\text{IE} \in E \cup C$  DO  $\alpha[\text{IE}] := 0$  //1. step
  FOR all  $\text{IE} \in q$  DO  $\alpha[\text{IU}] := 1$ 
  FOR all  $\text{IE} \in E$  such that there exists  $\text{IE}' \in E$  such that  $\text{sim}(\text{IE}', \text{IE}) > 0$ 
    DO  $\alpha(\text{IE}) := \pi_{\text{IU}}(\text{sim}(\text{IE}_1, \text{IE}) \cdot \alpha(\text{IE}_1), \dots, \text{sim}(\text{IE}_k, \text{IE}) \cdot \alpha(\text{IU}_k))$  //2. step
  FOR all  $c \in C$  such that there exists  $\text{IE} \in E$  with  $\text{rel}(\text{IE}, c) > 0$  DO
     $\alpha(c) := \pi_C(\text{rel}(\text{IE}_1, c)\alpha(\text{IE}_1), \dots, \text{rel}(\text{IE}_k, c)\alpha(\text{IE}_k))$  //3. step
  RETURN sort( $c$  with  $\alpha(c|c) \in C$  and  $\alpha(c) > 0$ )
END

```

Here, sort denotes the ordering of the arguments according to their computed similarity values.

This follows in principle the procedure of the linear weighted measures where weights are multiplied by local measures. However, more sophisticated amalgamation functions can be used, and one is not restricted to linearity.

The most important aspect is that not just similarity measures are computed; but at the end, the nearest neighbours are determined. The main properties of basic case retrieval nets are summarized as follows:

Advantages:

- Modular structure.
- Efficient retrieval.
- Effort dependent on the number of activated IEs in the query.
- Incremental extension if new cases come in possible.
- Missing values tolerated.

Disadvantages:

- High generation costs for the net.
- Problems with numeric attributes.
- For a high connection degree of the net (many IEs are similar to each other) much propagation to be performed.

Possible extensions to more general case retrieval nets:

- Subsymbolic computations.
- Rules.
- Lazy evaluations.

14.2.2 *Fish and Shrink*

This method is of different character than the previous one. It is aimed at retrieving cases that have a complex structure, such as a distinguished building. We also assume that there are not too many such cases. The method has two different specific elements that are combined:

- The first one is to introduce views, called aspects, on a case. Each view has its own aspect similarity measure. Views are easier to handle than the whole cases and break down the complexity of the task.
- The aspect measures can be combined using weights that can be chosen by the user at runtime. This means the global measure is not specified a priori.

The second element is concerned with retrieval itself. It says:

- If a case with a low similarity value to a query is found (is “fished”) then its neighbours are automatically not considered anymore, i.e., the search base is “shrunk”.

As an example, we consider someone who has to vote for a parliament and would like to have a candidate with opinions that are close to his or her views. First, we select a person randomly (fish). If this person is totally unacceptable he will be rejected. For selecting a second candidate it would be stupid to take persons that are quite similar to the first person; they are also omitted without detailed examination (shrinking).

The intention is to apply this in situations where the objects are quite complex and the similarity computation is fairly involved. Then the shrinking avoids many similarity computations.

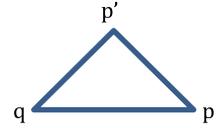
The case representation is not limited to simple attribute-value or even object-oriented representations. In fact, in certain complex situations such representations cannot represent the intended information and more involved representations are needed.

However, the representation methods in this section differ essentially from those that have been considered so far. Each way to look at an object gives rise to its representation method, the aspect. Therefore, the case representation is split into several aspects, each representing a complex property.

Such aspects are frequently available in knowledge management; see Chap. 21, Knowledge Management. Examples are financial, ecological and efficiency aspects. For each aspect a company usually has a specific department.

Next, we include some formal notation. CB is the case base. R is the set of all representations. In general, there is no uniform representation prescribed. Although, for instance, the whole car is considered, the weights can still be quite different. Sometimes attributes are negligible, which can be expressed as the view distance being zero. For instance, the relevance of choosing some material may be very small for pricing but high for recycling.

The set of aspects is denoted by $A = \{a_1, \dots, a_n\}$. One can consider an aspect as an index of a case. For each aspect a_i of a case c , there is a mapping $CB \rightarrow \mathfrak{R}$; the result is denoted by $a_i(c)$; it is the representation form of c .

Fig. 14.2 Triangle inequality

Here we use distance functions. We first define distances for aspect representations. Aspect measures are not defined for parts of objects but for whole objects. One can say that they emphasize some parts of the object description.

The aspects are connected with views, which will be described in the next definition.

Definition 14.4

- (i) For each aspect $a_i \in A$ there exists an *aspect distance function*

$$d_i : CB \rightarrow [0, 1], \quad d_i(a_i(c), a_i(p)) \in [0 \dots 1]$$

(in short: $d_i(c, p)$ for cases c and p).

- (ii) A *view* is some weight vector $\omega = (\omega_1, \dots, \omega_n)$ with $\omega_1 + \dots + \omega_n = 1$.

The set of views is V . Views will be a user-defined part of the query.

- (iii) A *view distance* for two cases and a view is a function

$$VD : CB \times CB \times V \rightarrow [0, 1].$$

- (iv) A case c is *view neighbour* of a case p relative to a view distance VD if $VD(c, p, \omega) :< \tau$ holds, where τ is a user-defined threshold.

The number ω_i reflects the importance of the aspect a_i , as in any other weighting. More examples of view distances:

- (a) $VD(c, p, \omega) := 1/n(\omega_1 d_1(c, p) + \dots + \omega_n d_n(c, p))$ average.
 (b) $VD(c, p, \omega) := \text{Max}(\omega_i d_i, (c, p))$ pessimistic.
 (c) $VD(c, p, \omega) := \text{Min}((1 - \omega_i) d_i, (c, p))$ optimistic.

In applications, a user-chosen importance can define view distances.

The importance of the triangle inequality motivates the choice of the distance functions as measures: The point is that the method requires a restriction: For the distance functions, the triangle inequality we assume this as in Fig. 14.2.

Both inequalities are equivalent. Here we use the inequality in the second form. This gives rise to a representation of the cases in a net with aspect distances as annotations to the edges. We observe here an analogy to case retrieval nets.

$$d(q, p) \leq d(q, p') + d(p', p),$$

$$d(q, p') \geq d(q, p) - d(p, p').$$

Next, we turn to the retrieval functions. The two major actions are:

- Fish: Compute the similarity between the query and a test case.
- Shrink: Make use of the test case to get information about the similarity of other cases without computing the similarities.

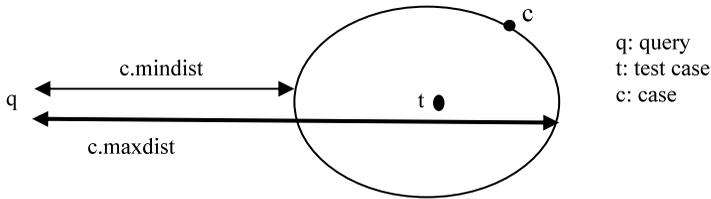


Fig. 14.3 Minimal and maximal distances

After selecting a view distance VD_{user} (provided by the user), a first test case is selected randomly for finding an answer to a query. This has led to the term fishing. The test case t is compared with the query with respect to the view. The idea now is that the study of the test case t leads to a reduction of the search space, as indicated in the motivation.

Definition 14.5

- (i) The test distance is $VD_{user}(q, t, \omega)$.
- (ii) For a view neighbour c , the base distance is $VD_{user}(t, c, \omega)$.

The aspect distances for the test case are denoted as $testdist$. One computes them using the triangle inequality, which allows estimating the aspect distances between the query and the not-yet tested cases c by giving the minimal and maximal expected distances $c.mindist$ and $c.maxdist$, as shown in Fig. 14.3.

A query contains a case q (as usual), and in addition a view V , i.e., a weight vector. This allows the user to express an importance at runtime after the query is formulated. Now an essential idea is to precompute the aspect distances between certain test cases (not necessarily all cases). This is the method of choice if the distance computation requires much effort.

In addition, we assume now that the user has a certain view distance VD_{user} .

The idea is now to compute for the selected test cases the distances with respect to VD_{user} a priori. To make use of this, one takes the following arguments, summarized in a rough sketch of the retrieval algorithm:

- (1) Determine for each case in CB a distance interval (initially $[0 \dots 1]$).
- (2) Select a test case t from CB (fishing) and determine a view distance between q and t (much effort).
- (3) Compute for each case in CB the new (smaller) distance interval by using the triangle inequality.
- (4) Iteration, until the intervals are sufficiently small, as user-defined.

The min and max distances are dynamic and will be updated by the algorithm. When more test cases are used, the interval $[mindist, maxdist]$ will be reduced, which is called “shrinking”. This takes place in Step 3.

In the next algorithm, we will introduce some auxiliary predicates. The cases in the case base have now different minimal distances.

There are two notions that control the search:

- The OK predicate: Indicating termination;
- The precision line PL: Indicating which cases should be removed next. This is a dynamic predicate.

Both concepts deal with the demanded accuracy of the solution and are provided by the user. All cases with $\text{mindist} = \text{PL}$ are now candidates for the next test. This line is again dynamic and will be moved during the algorithm.

The process terminates if the approximation seems to be good enough as indicated by the predicate OK. The precision line PL determines which cases are eliminated; it will be offered to the user. The algorithm “fish and shrink” makes use of these notions; it is given in pseudocode next.

```

Input: base  $CB$ , query  $q$ , view  $\omega$ 
VAR:  $t$  (test case)
 $c.\text{mindis}$  (expected minimal distance) of the case  $c$ ,
 $c.\text{maxdis}$  (expected maximal distance) of the case  $c$ ,
 $c.\text{basedis}$  (base distance) of the case  $c$ ,
testdis (test distance)
PL (precision line)
FOR all  $c \in CB$ 
DO  $c.\text{mindis} := 0$   $c.\text{maxdis} := 1$ 
END
  WHILE NOT OK DO
    Move precision line PL
    Choose (fish) an object  $t$  on the precision line
    testdis :=  $\text{VD}(q, t, \omega)$  (*view distance*)
     $t.\text{mindis} := \text{VD}(q, t, \omega)$ 
     $t.\text{maxdis} := t.\text{mindis}$ 
    FOR all  $c \in CB$  where  $c$  is a view neighbour of  $t$  AND  $c.\text{mindis} \neq c.\text{maxdis}$ 
    DO  $c.\text{mindis} := \max(\text{testdis} - \text{basedis}, c.\text{mindis})$  (*Shrink*)
     $c.\text{maxdis} := \min(\text{testdis} + \text{basedis}, c.\text{maxdis})$ 
  END
END

```

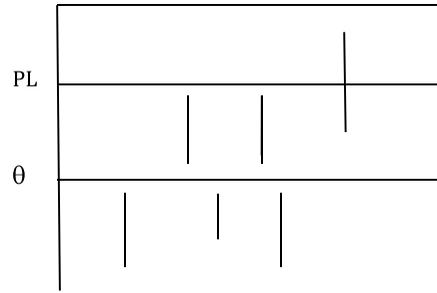
By suitable choices of OK and the precision line PL, different retrieval tasks can be satisfied. There are different termination criteria OK, depending on the intention, for instance, returning the nearest or the k -nearest neighbours. In Fig. 14.4, two cases qualify.

The method can either be applied to all cases that are more similar than a given threshold θ or deliver the k best cases.

OK is a predicate that indicates that user demands are satisfied. A simple way to do this is by taking the threshold θ for the required similarity to the query.

The precision line PL is defined as the highest possible similarity of cases for which it is not yet decided whether they fulfil the retrieval criterion. The precision line PL is responsible for the shrinking.

Fig. 14.4 All cases better than threshold θ



The main properties of Fish and Shrink are summarized as follows:

Advantages:

- Flexible distance computation to the query using views.
- Different retrieval tasks can be considered.
- Efficient because many distance computations can be saved.
- The method can be used as “Anytime Algorithm”. The retrieval can be interrupted at any time and still deliver useful results, although the retrieval criterion might have not yet been completely fulfilled.
- The algorithm is incremental in the sense that it can to some degree deal with changing case bases.

Disadvantages:

- Aspect distances have to be computed a priori.
- Distance function has to satisfy the triangle inequality.

Recommendation:

- Useful if very complex distance measures are present (for instance, complex object or graph representations).

As an example we mentioned the domain architectural design. This domain includes many dimensions, for instance, installation of water and electricity, which have their own views and where special cases play a role. This requires a complex similarity computation because of the very involved topological structure of the objects.

14.2.3 PROTOS: Another Two-Step Retrieval

The retrieval in PROTOS is another two-step retrieval approach dealing with classification. The purpose is to limit the search space rather than maintaining high accuracy in classification tasks. It is based on the combination of two case bases.

The first base is a prototypical case base where one describes each category of classes. The concept of a prototypical case base is to rely on artificial cases that summarize a class. For example, instead of using specific attributes like age in years, age is a range of attributes that allows values in intervals (e.g., young adults or $19 \leq \text{age} \leq 29$).

The second case base is comprised of real cases. With each classification category, a set of real cases is associated. This is how one can add cases indefinitely to the case base without slowing down the overall retrieval. The first step in PROTOS will be always equally fast.

- (1) First step: Surface features of a new query are compared to a case base of prototypical cases.
- (2) Second step: The new query is compared to the most prototypical case among the real cases that are instances of the prototype selected in the first base. Note that this second comparison is done at a more specific level of abstraction where all attributes are now used. It is usual that this comparison results in a poor match. Then one uses the differences pointed out in this comparison with associations in the memory to find a better match within the real cases.

The associations link cases on a certain attribute by pointing to a case with a different value for that attribute. With such associations, simply following the link will lead to a better match. One learns these associations and keeps the system from repeating misclassifications.

14.2.4 Fuzzy Retrieval

This section deals with situations where local acceptance with respect to a feature can be expressed by a fuzzy predicate. The potential benefit brought by fuzzy retrieval relies on not limiting the aggregation of local similarities to methods that are subject to additivity axioms.

Aggregation methods such as the weighted mean or Euclidean distance used in non-fuzzy similarity measures allow for a small amount of very relevant features that are valued on one extreme (i.e., either similar or dissimilar) to be compensated by a large amount of not so relevant features valued at the other extreme.

Fuzzy integrals represent a class of functions that use importance measures that are minimally required, forcing the ordering of features in such a way that, in addition to providing an average, they guarantee that all relevant features participate in the average. This can be complemented by the use of fuzzy queries.

14.2.5 Comparison

In Table 14.1 we summarise of the properties of the discussed retrieval methods.

14.2.6 Reducing the Search Space and Preprocessing

For any case base, one needs in principle to compute the similarities between the problem and every case in the base and then sort them by similarity. This can be very time consuming. Reducing the search space is a preprocessing for retrieval.

Table 14.1 Comparison

Method	Restriction w.r.t. similarity	Recommended for
Sequential search	No	Small case bases, simple similarity measures
Two-level search	No	Large case bases
Voronoi diagrams	Numerical similarities	Geometric case structures
kd-tree	Ordered domains and others	Few attributes, many cases
Retrieval nets	Problems with numeric values, monotonicity needed	Case completion wanted, few attributes, large case bases
Fish and shrink	Triangle inequality needed	Complex similarities, small case bases
Protos	Classification	If many misclassifications occur
Fuzzy retrieval	No	If weights are not clear

The purpose of some types of preprocessing is to filter out some cases. The goal is to find a compromise between having few cases for searching and not losing good solutions. This refers to Chap. 9, Adaptation, where learning essentially meant “learning to forget”.

There are different possible ways to reduce the case base. Some general possibilities are:

- (1) Sampling: Store only some samples of the cases. Sample selection is a large problem. A typical approach is to choose typical examples for the solutions.
- (2) Boxing: Create boxes that enclose clumps of examples that are of the same type. One can then store the box’s centre and dimensions, rather than storing every example in the box.
- (3) Counting how often a case was called and investigate whether it may be redundant.
- (4) Using rules for hard and weak constraints.
- (5) Using footprints, considered below.

Hard constraints definitely exclude cases while weak constraints reduce their likelihood. Both constraint types can be formulated with rules; the preconditions are the constraints and the actions generate the likelihood as a number. The rules can be combined and the combinations influence the numbers. Cases with similar likelihood can be grouped into boxes. The violation of constraints can have different reasons, for instance:

- The problem is dynamically changing and constraints arise over time while the case base remains the same.
- The case base is taken from somewhere else where the same conditions do not hold. Now we will investigate the use of rules a little more closely.

An important method for this is presented in Chap. 9, Adaptation. Adaptation basically means obtaining new cases by applying adaptation rules to cases already in the case base. Thus, the application of such rules allows deleting cases from

the base that are generated by the rules. Essentially, this is a shifting from the case base container to the adaptation container. When adaptation is used, it is guaranteed that no solutions are lost.

For example, sometimes a decision can be made based on certain attribute-values alone, without investigating any further attributes. To make use of this idea the following has to be done:

- (1) Identify such attributes and decisions.
- (2) Analyse the cases and select those cases in which such attributes and the decisions occur.

The attributes to be found are usually discriminating opportunities for the decision. The attributes are found in the problem part and the decision is found in the solution part of the case. One cannot neglect the decisions because the attributes may occur in other cases with different decisions.

Examples are:

- If your age is over 65 and you are a resident of our city then you are entitled to a reduced price for tickets for public transportation, and otherwise you are not.
- If the car loans, housing loans, or other credit card payments are below a certain limit then the customer is entitled to a gold card if and only if the applicant has a yearly income over a certain limit and has no previous record of unpaid balances.

In both situations, one can easily define rules for the decision. Checking the attributes entails the same process in CBR and in rule applications. However, the application of the rules makes the similarity search redundant.

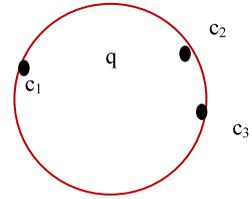
14.2.6.1 Footprints

We discussed this in detail in Chap. 9, Adaptation. Here are some additional views. This kind of retrieval is based on two searches on two separately edited subsets. The first search identifies a reference case that is similar to the query. This case acts as an index to the complete case base, and the second search locates the best available case in the region of the reference case. This two-step method has in common with the Fish and Shrink and Protos approaches the fact that there are also two steps where the first step deals with aspect similarities.

Here, in the first step one selects a compact competent subset of the case base to retrieve a case that is similar to the query. This case acts as a reference point for the full case base. The reference case may not provide a solution to the query. In the second step, one compares the query with the cases similar to this reference case, and the most similar one to the query is retrieved.

The definition of the footprint set uses the technique mentioned above when working with adaptation. This means one searches for the best solutions obtained after adaptation. More technical details are given in Chap. 9, Adaptation.

Fig. 14.5 Two-dimensional example



14.3 Similarity Diversity

Suppose we have the retrieved k cases $c = \{c_1, \dots, c_k\}$ as the k -nearest neighbours for a query q . The number k may be too large to be useful, in particular, if these neighbours are very close to each other.

For example, if two offered cars differ only in colour, one being light blue and the other being medium blue, then one would not offer both. Note that this is particularly useful for applications where solutions require variety.

If there are several nearest neighbours, the question is: Which neighbours should we select? Consider in Fig. 14.5 the following two-dimensional example with Euclidean distance.

For the query q , there are three nearest neighbours c_1 , c_2 , and c_3 . Suppose we are restricted in that only two of them can be mentioned. It is obvious to mention just $\{c_1, c_2\}$ or $\{c_1, c_3\}$ because these give the customer a much better choice. The points $\{c_2, c_3\}$ are very close together and will not provide enough diversity to the user.

One may ask why the system provides such diverse nearest neighbours at all. The answer is that the cases do not contain all attributes that are of interest to the customer, and therefore the system cannot distinguish between certain cases.

As a consequence, we may be interested in the similarity distribution of the cases. To study this we introduce the CBR analogues to mean and variance in statistics.

Suppose a set $C = \{c_1, \dots, c_k\}$ is given.

Definition 14.6 The average similarity of C is

$$\text{sim}_{\text{av}}(C) = 1/k \sum_{i=1}^k (\text{sim}(p, c_i)).$$

The opposite of similarity is dissimilarity, which plays a role here. In order to describe how dissimilar a whole set is, we introduce the notion of diversity.

Definition 14.7 The diversity of C is

$$\text{diversity}(C) = \frac{\sum_{i=1}^n \sum_{j=i}^n (1 - \text{sim}(c_i, c_j))}{n(n-1)/2}.$$

This means the diversity measures the average dissimilarity in the set. When the cases are represented as graphs, the notion of group similarity is useful. This notion

measures the average similarity of a case set in terms of the graph representation and can be regarded as the dual of diversity.

In the first view, there is an optimization problem involved: Find for a set of cases a subset of a fixed size n that has the highest diversity. However, what we have here is a trade-off between a high similarity to the query and a high diversity among the nearest neighbours. Many nearest neighbours that are also close to each other are not very useful, but deleting cases may omit desired solutions.

This applies also if one specifies the query vaguely. Again, there is in addition the question that the utility of the case may depend on unknown criteria too. This suggests again considering alternatives that are very different from each other although some criteria are violated.

The refined *diversity problem* is now to offer a set of alternatives that are not only maximally distant from each other but are also close to the query.

This leads mathematically to the following optimization problem:

- (1) Maximize the similarity to the query.
- (2) Minimize the similarity of the retrieved objects to each other.

This gives rise to a quality function for the retrieved set C of cases that guides the selection of cases. Now we assume that some set $S = \{s_1, \dots, s_k\}$ has been selected already. The quality function takes into account the query, the actual case, and the already selected cases.

Definition 14.8

$$Quality(q, c, S) := \text{sim}(q, c) \cdot \text{relativeDiversity}(c, S),$$

$$\text{relativeDiversity}(c, S) := \begin{cases} 0 & \text{if } S = \{\}, \\ \sum_{i=1}^k 1 - \text{sim}(c, s_k) & \text{else.} \end{cases}$$

This gives a way to select cases iteratively for building S , which is ultimately the set of selected nearest neighbours:

- Initially one selects the case with the highest similarity to the query. Here $S = \emptyset$ holds.
- In an iteration step, one selects the case with the highest combination of similarity to the query and diversity with respect to the actual set S .

It should be remarked that the above definition of quality is not the only possible one. There are several alternatives that are, however, all in the same spirit.

If one compares similarity with relative diversity, one difference is obvious:

- Similarity is more local with respect to cases; the similarity between two cases does not depend on the similarity between other cases.
- Diversity is more global with respect to cases; it always considers a larger set of cases.

14.4 Which Retrieval Method Should I Use?

For special purposes it may be useful to choose a special retrieval method from a choice in a CBR tool or implemented on your own. Several aspects influence such a decision. Instead of giving precise recommendations, we suggest a number of questions that the user should investigate.

First, the data structures may prohibit or require the use of some methods; as for instance, the use of kd-trees is restricted. The next step is to investigate the characteristics of the application with respect to the abilities of the methods and tools. Here, one may have to compromise between complex indexing and efficient retrieval. This depends on the structure of the cases, the number of cases and how they are represented.

Then one investigates how much the best solution is desired and whether a sub-optimal solution is sufficient. This again reduces the choice of retrieval methods.

Another question is, which aspects of my data representations are unsatisfactory and how can I improve them? This will require a preprocessing method, leading to a more appropriate data structure or case representation. One has to investigate whether the effort of preprocessing pays off with respect to the reduced retrieval effort.

Another view is with respect to tool support. Suppose that you have to evaluate a large set of examples. Then it is quite time consuming to enter each problem and retrieve each solution individually. A tool could do this for you. If it is not available, you have to write a procedure (in this case a loop) for this purpose.

14.5 Tools

All CBR systems have some retrieval functions build in. One has developed special tools for special applications and special representation forms like texts or images.

The tool jColibri has also methods to deal with diversity; see Recio-García and Díaz-Agudo (2004).

14.6 Chapter Summary

First, we considered two advanced retrieval methods: Case Retrieval Nets and Fish and Shrink. Both require a special case representation form. In the case of retrieval nets, the queries and the cases are incrementally completed. Fish and Shrink deals with very complex situations that have different aspects and complex views.

The two-step retrieval methods are complemented with the PROTOS approach. In terms of aggregation of values for retrieval, the fuzzy retrieval is not limited to additivity axioms.

There are several methods presented to improve retrieval by reducing the case base. Diversity deals with the problem of there being too many nearest neighbours, and we showed how to reduce this number.

14.7 Background Information

Case retrieval nets were introduced by Mario Lenz; see Lenz and Burkhard (1996) and Lenz (1999). The Fish and Shrink method was invented by J. Schaaf in his dissertation; see Schaaf (1996). This method originated from working with complex objects, in particular, architectural buildings. These problems may not occur often but the method is still of interest. Chakraborti et al. (2006) propose a variation to speed up case retrieval nets.

Fuzzy retrieval through fuzzy integrals has been first discussed in Weber-Lee et al. (1995) and then extended in Wang and Yeung (2000). Well-known algorithms for fuzzy integrals are the Sugeno integrals. Reducing the case base for improving retrieval was also a topic of Chap. 10, Evaluation, Revision, and Learning, and was also discussed in Chap. 9, Adaptation. Using rules for this is shown in Chen and Wilkinson (1998).

The retrieval approach in PROTOS has been described in the following publications, where one can learn further about it: Bareiss et al. (1988), and Porter et al. (1990).

Diversity was introduced by Smyth and McClave (2001); see also McSherry (2002). Diversity plays a major role in recommender systems, where the customers should be offered an optimal and small set of products.

14.8 Exercises

Exercise 1 Describe a complex building in detail and define aspects from the viewpoint of the architect, the civil engineer and the buyer.

Exercise 2 Describe the retrieval for the beach example using sequential retrieval and comment on it.

Exercise 3 In the domain of used car sales, describe a situation where there are eight different cars of (almost) equal similarity to a query but you want to show only two of them.

- (a) Which ones would you choose?
- (b) Describe the underlying principle with respect to diversity.

Exercise 4 Build a BCRN for short descriptions of restaurant types, e.g., Chinese, Japanese. Use two different customer types of your choice. Describe the steps to be taken for retrieval.

Exercise 5 Build two flat case bases for classifying foods using nutrition facts, one of them using only attributes calories, fat, protein, and carbohydrates, using the other nutrition facts available. Demonstrate the different retrieval results for the classification problems.

References

- Bareiss R, Porter B, Weir CC (1988) Protos: an exemplar-based learning apprentice. *Int J Man-Mach Stud* 29:549–561
- Chakraborti S, Lothian R, Wiratunga N (2006) Fast case retrieval nets for textual data. In: Roth-Berghofer TR, Göker M, Güvenir HA (eds) *ECCBR 2006: advances in case-based reasoning. 8th international conference, Fethiye, Turkey, September 2006. Lecture notes in computer science (lecture notes in artificial intelligence)*, vol 4106. Springer, Berlin, p 400
- Chen H, Wilkinson L (1998) Case match reduction through the integration of rule-based and case-based reasoning procedures. In: Aha DW, Daniels JJ (eds) *Case-based reasoning integrations: papers from the 1998 workshop. Technical report WS-98-15. AAAI Press, Menlo Park*, p 33
- Lenz M (1999) *Case retrieval nets as a model for building flexible information systems. Dissertation, Humboldt University*
- Lenz M, Burkhard H-D (1996) Case retrieval nets: basic ideas and extensions. In: Goerz G, Hoell-dobler S (eds) *KI-96: advances in artificial intelligence. Lecture notes in computer science*, vol 1137. Springer, Berlin, p 227
- McSherry D (2002) Diversity-conscious retrieval. In: Craw S, Preece A (eds) *ECCBR 2002: advances in case-based reasoning. 6th European conference, Aberdeen, UK, September 2002. Lecture notes in computer science (lecture notes in artificial intelligence)*, vol 2416. Springer, Berlin, p 219
- Porter B, Bareiss R, Holte R (1990) Concept learning and heuristic classification in weak-theory domains. *Artif Intell* 45:229–263
- Recio-García JA, Díaz-Agudo B (2004) *An introductory user guide to jCOLIBRI 0.3. Technical report 144/2004, Universidad Complutense de Madrid*
- Schaaf J (1996) Fish and shrink: a next step towards efficient case retrieval in large scaled case bases. In: Smith I, Faltings B (eds) *EWCBR-96: advances in case-based reasoning. Third European workshop, Lausanne, Switzerland, November 1996. Lecture notes in computer science (lecture notes in artificial intelligence)*, vol 1168. Springer, Berlin, p 362
- Smyth B, McClave P (2001) Similarity vs. diversity. In: Aha DW, Watson ID (eds) *ICCBR 2001: case-based reasoning research and development. 4th international conference on case-based reasoning, Vancouver, BC, Canada, July/August 2001. Lecture notes in computer science (lecture notes in artificial intelligence)*, vol 2080. Springer, Berlin, p 347
- Wang XZ, Yeung DS (2000) Using fuzzy integral to modeling case-based reasoning with feature interaction. In: *2000 IEEE international conference on systems, man, and cybernetics*, vol 5. IEEE, Los Alamitos, p 3660
- Weber-Lee R, Barcia RM, Khator S (1995) Case-based reasoning for cash flow forecasting using fuzzy retrieval. In: Veloso MM, Aamodt A (eds) *ICCBR-95: case-based reasoning research and development. First international conference on case-based reasoning, Sesimbra, Portugal, October 1995. Lecture notes in computer science (lecture notes in artificial intelligence)*, vol 1010. Springer, Berlin, p 510

Chapter 15

Uncertainty

15.1 About This Chapter

In this chapter we discuss relations between CBR and some uncertainty methods. The main topics are rough sets and fuzzy sets. These concepts occur frequently in applications. Of special interest is the relation to similarity.

15.2 General Aspects

Most descriptions of situations and problems are not quite exact. They contain a number of uncertainties and these can be of different natures. In this chapter we consider some concepts of and methods for such uncertainties. This means we consider representations weakening the classical true-false paradigm. It is one of the advantages of CBR that due to its basic principles it is able to deal with such extended problems. In particular, the roles of the process model and the knowledge containers remain unchanged. First we introduce general aspects of uncertainty and then we investigate some example types.

General tasks that we consider are:

- We want to measure uncertainty;
- We want to minimize uncertainty;
- We want to transfer the results into binary yes-no decisions.

Two most prominent uncertainty methods are fuzzy sets and probabilities. Both occur frequently in everyday problems as well as in industrial tasks. They have also a close relation to similarity measures. As a consequence, both approaches have much in common with CBR. Fuzzy sets are central in this chapter. Probabilities are the subject of Chap. 16, the next chapter. A general method for dealing with uncertainty is rough sets.

The presence of uncertainty in statements (i.e., they are not simply either true or false) is due to several reasons:

- (a) Some description parts are missing;
- (b) One does not know precisely the value of some magnitude;
- (c) One wants to express degrees of a property only;
- (d) One has only a probability of an event;
- (e) One makes formulations in a vague manner.

The basic difference between the uncertainties handled by fuzzy methods and by probabilities is:

- Fuzzy sets consider situations where something that occurs can only be expressed approximately by degrees and not exactly.
- Probabilities consider situations that occur sometimes but not always.

CBR has itself an inherent uncertainty. The uncertainty is reflected in the similarity measures. However, the arguments of the similarity measures have been of a crisp character so far. This will be changed now by allowing imprecisely described objects as arguments to the measures.

15.3 Uncertainty Concepts and Methods

15.3.1 Rough Sets

First, we will introduce the concept of rough sets, which is useful for all situations where uncertainty occurs. It has already been mentioned and motivated in Chap. 2, Basic CBR Elements. The method is not restricted to special approaches and is of use to CBR too.

We consider a universe U and assume that the uncertainty is given by a binary relation “ \approx ” over U . This relation is called an *indiscernibility relation*. The intention is that in our present situation we cannot distinguish objects a and b that are in the relation $a \approx b$.

There are mainly three possible reasons for indiscernibility:

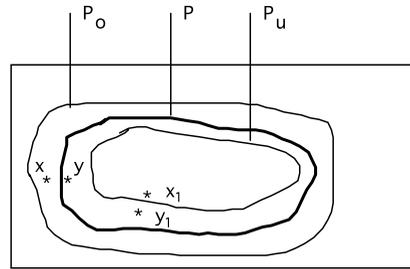
- (a) Tolerances: Small distances cannot be seen.
- (b) Missing values of attributes: Some objects differ because they have different values in certain attributes but these values cannot be seen.
- (c) Missing attributes: If whole attributes are invisible the same problem as the one in (b) arises.

In all three cases \approx is reflexive and symmetric; in the latter two it is even transitive, that is, it is an equivalence relation.

Such situations happen quite often in reality. The problem comes up if one has to make a 0–1 decision despite the lack of information. Such a decision divides the set of situations into two sets:

- (1) The 1-set: In these situations the decision is “yes”.
- (2) The 0-set: In these situations the decision is “no”.

Fig. 15.1 Rough sets



These sets are unknown to the decision maker. However, we assume that there is not complete ignorance present and some knowledge about the decisions exists. This knowledge is assumed to be of the form:

- (1) In some situations one knows that the answer is *yes*.
- (2) In some situations one knows that the answer is *no*.

This gives rise to two sets that are disjoint but do not cover the entire decision space.

This will now be discussed in an abstract setting. We consider some predicate $P(x)$ over a universe U that represents a subset of U . The decision to be made is whether some element has or does not have the property P .

The relation \approx motivates the following:

Definition 15.1

- (i) The *lower approximation* of P is

$$P_l = \{x \in U \mid \text{for all } y \text{ with } x \approx y : y \in P\}.$$

- (ii) The *upper approximation* of P is

$$P_u = \{x \in U \mid \text{there is } y \in P \text{ with } x \approx y\}.$$

The elements of P_l are surely elements of P , the elements of P_u are possibly in P , and the elements in $P \setminus P_u$ are surely not in P .

The set $P_u \setminus P_l$ is called the *uncertainty area*. The underlying idea is to notify the user about these sets. Additional attention has to be paid to the uncertainty area. For instance, questions may be asked to obtain more information about the situation.

Because decisions about elements in P_l and elements not in P_u are certain, the rough set method can be regarded “to be on the safe side”. For elements in the uncertainty area one has to look for further information in order to make a decision. The goal is of course to make the uncertainty area as small as possible. One should observe that the rough set method gives an honest answer: It tells you where one is uncertain. One sees this in Fig. 15.1.

Here we assume $x \approx y$ and $x_1 \approx y_1$. Now we discuss this approach in the context of CBR. The nearest neighbour (i.e., in the given case base) is not always sufficient for providing an acceptable solution. For instance, the nearest neighbour may

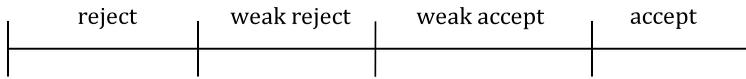


Fig. 15.2 Uncertainty regions

still have a low similarity to the query. On the other hand, a case which is not the nearest neighbour may be sufficiently close. For this purpose one can introduce two thresholds α and β , $0 < \alpha < \beta < 1$ with the intention as follows:

- If $\text{sim}(\text{newquery}, \text{caseproblem}) < \alpha$ then the case is not accepted;
- If $\text{sim}(\text{newquery}, \text{caseproblem}) > \beta$ then the case is accepted.

This has been illustrated in Fig. 6.4 in Chap. 6, Basic Similarity Topics. It partitions the case base for the actual problem into three parts: Accepted cases, unaccepted (rejected) cases, and an *uncertainty set*. In Definition 15.1 we saw a further refinement.

In many cases, one can divide the uncertainty region further, as in Fig. 15.2.

In Chap. 2, Basic CBR Elements, we discussed that one can introduce negative cases which correspond to the rejected cases. The advantage of introducing weak rejects and weak accepts is additional information to the user, who can thus react accordingly.

In the general situation where we deal, for instance, with demands and products, the same partition arises. In practical situations one has to make a decision because one has to do something and cannot do it partially, for example, 70 %. For this reason the uncertainty area has to be split again into two yes-or-no areas in which the decision has to take place. The role of the uncertainty area is here to give the user an additional comment because in practice the user can take additional precautions or actions if informed.

An alternative way is (as indicated in Chap. 2, Basic CBR Elements) to introduce two sets of cases, CB^+ (corresponding to accepted cases) and C^- (corresponding to rejected cases).

15.3.2 Fuzzy Sets

Fuzzy sets are a popular way to represent uncertainty. Basically, they extend the view of rough sets in the way that boundaries of whole sets may not be exactly known, or are vague in nature. This extension is, however, deeper in several respects.

It does say not only that we do not know sets precisely, but rather that certain sets do not exist in a 0–1 fashion. This results from the fact that certain properties defining sets are by their very nature given in degrees. This means the property is not of a 0–1 nature. It leads to the fact that a membership is given in degrees too.

When discussing degrees we encounter the same distinctions as in the analysis of similarity and utility:

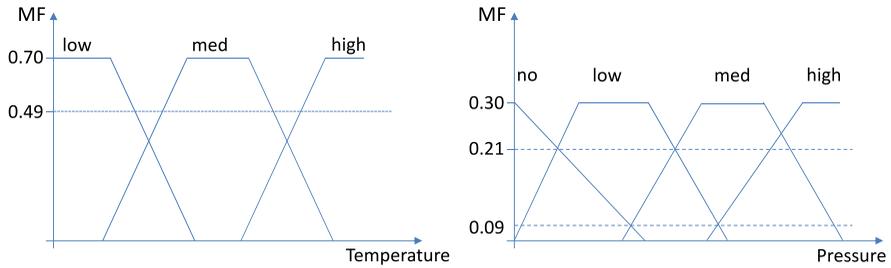


Fig. 15.3 Fuzzy degrees

- (a) Degrees can be relational or functional. Relational degrees occur when we say “Object x has the property P to a higher degree than object y ”. Functional degrees are expressed in terms of numbers.
- (b) Degrees can be objective or subjective. Objective degrees arise from a (formal) model while subjective degrees reflect the opinions of agents.

As already mentioned, not all decisions can be done in degrees. If I am told “accelerate the car” I can do it in degrees. But if someone wants me to buy a ticket I cannot buy the ticket 70 %: Either I buy the whole ticket or I do not. For this purpose a transformation back into 0–1 logic may be necessary.

Therefore, there are two operations:

- (1) Introduction of membership degrees, called fuzzification;
- (2) Converting back to the 0–1 situation, called defuzzification.
- (3) For example, we are given two attributes, Temperature and Pressure, that have numerical values. Now two attributes Temp and Press with symbolic values are introduced:
- (4) Temp: {low, medium, high} and Press: {no, low, medium, high}.
- (5) In the fuzzy approach these attributes are taken as linguistic expressions. In the next step, “fuzzification” membership functions (MFs) are introduced. They determine the degree of membership of a temperature to low, medium or high, and the same takes place for pressure. In the example the membership functions are taken as piecewise linear. We see this in Fig. 15.3.

The picture indicates some typical properties of such representations.

- (a) A measured temperature or pressure can belong with some degree to two sets. It can even belong to them with the same degree.
- (b) The membership functions are of a special form.

What is the justification for this? Do we have fuzzy degrees for the combination of high degree and low temperature? Belonging to two fuzzy sets reflects uncertainty such that one cannot say that something belongs to one set only.

The choice of the membership function is somewhat problematic. It goes back to the linguistic level, on which experts have a motivation for choosing the predicates. It comes from the desire to perform some actions like “stop the car” or “speed up”

that depend on temperature and pressure. When determining the degrees, one has in mind that they should reflect the influence on the intended action. This introduces again an uncertainty into the problem. The choice of the functions reflects the opinion that should they reflect the influence well.

This is parallel to the concept of utility discussed several times in this book. Utility is influenced by several factors that are combined into a number that is an estimation of the overall utility. Therefore, it is not wrong to regard utility as a special instance of fuzzy set theory. The combination of fuzzy functions is discussed in the next section.

At this point the question that comes up is, how is it justified to call fuzzy sets an uncertainty approach? There are many scientific areas where degrees are used that are perfectly exact. We discuss this in the next section.

15.3.3 Basic Elements in Fuzzy Set Theory

Fuzzy sets formalize the idea of graded membership. For this purpose fuzzy membership functions are introduced. They assign values to elements of an arbitrary domain U with respect to a certain property where one does not accept just “yes” or “no” assignments.

Definition 15.2 A fuzzy membership function is of the form

$$\mu : U \rightarrow [0, 1].$$

Membership functions μ are associated with *fuzzy subsets* (or predicates) P of U ; the reference to P is denoted by μ_P . $\mu_P(a)$ is called the degree with which a has the property P .

The next point to investigate is fuzzy reasoning. Reasoning in this area takes place in the form of applying rules. One starts with so-called linguistic rules. They are nothing else than the ordinary rules introduced in Chap. 9, Adaptation. They are of the form

$$\phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_n \rightarrow \psi$$

between expressions ϕ_i where ψ can be an action. These expressions formulate opinions of experts about how they would proceed.

After fuzzification the problem arises of how to work with the membership functions computationally. In this case the ϕ_i and ψ are replaced by the values of their membership functions.

As a first step we consider the preconditions of the rules. Because the expressions are combined with logical connectives, we need to compute the values of logical combinations given by the connectives. This means answering the question, “How do we compute the fuzzy analogue of set-theoretic operations”? This can be done in different ways.

There are two basic operations that allow formulating the analogues of classical set-theoretic operations on fuzzy sets. These are composition operators called t-norms and co-t-norms (corresponding to conjunction and disjunction).

These operators are described on an abstract level by certain axioms.

Axioms of t-norms $f(x, y)$ intended to compute $\mu(A \cap B)$ and the conjunction:

- (T1) $f(x, y) = f(y, x)$
- (T2) $f(x, f(y, z)) = f(f(x, y), z)$
- (T3) $x \leq x', y \leq y' \Rightarrow f(x, y) \leq f(x', y')$
- (T4) $f(x, 1) = x$

Axioms of co-t-norms $f(x, y)$ (intended to compute $\mu(A \cup B)$ and the disjunction):

- (T1), (T2), (T3) and
- (T4*) $f(x, 0) = x$

One easily verifies that for the classical Boolean case the conjunction and the disjunction satisfy the axioms of t-norms and co-t-norms, respectively. The mostly used t-norms are $f(x, y) = \min(x, y)$ and $f(x, y) = x \times y$, and popular co-t-norms are $f(x, y) = \max(x, y)$ and $f(x, y) = x + y - x \times y$. In both cases they are extensions of the traditional Boolean operations.

For our purposes two observations are important:

- (1) Norms and co-norms are symmetric, i.e., for complex fuzzy predicates there is no direct way to express importance or degree of influence of the constituents. The way fuzzy logic deals with this problem is by defining other fuzzy sets which cover the rest of the domain and occur in different fuzzy rules.
- (2) The membership functions identify all arguments with value 0; there are no negative degrees.

Next, we consider negation and implication. On the membership level, negation is just a numerical function, for instance, $1 - x$.

The situation for implication is more involved. The implications are not seen in the pure logic sense but are intended to represent conditions of an action. There are different ways to formalize implications. The implications are stated in the form:

$$\phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_n \rightarrow \psi.$$

The preconditions state the degrees of memberships of the terms. Now an essential difference with classical operators enters the scenario: The preconditions are satisfied to a certain degree only. As a consequence, the resulting action is also performed only to some degree.

There are several ways to formulate such a rule. A popular rule is the Mamdani rule.

Definition 15.3 The Mamdani computation operates as:

- The membership function of the conjunction is computed using a t-norm.

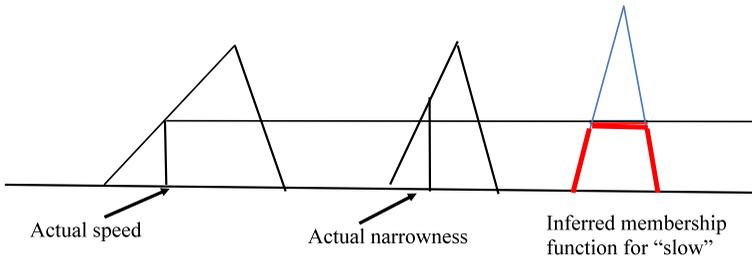


Fig. 15.4 A rule

- The implication is considered as a conjunction $\phi_1 \wedge \phi_2 \wedge \dots \wedge \phi_n \wedge \psi$.

This allows computing the value of the implication. If the consequence is an action, it says to what degree the action has to be performed. For example, consider the linguistic rule: If temperature is low and road is narrow then speed is slow. This is illustrated in Fig. 15.4. Again, all membership functions are piecewise linear.

The result of the implication is a membership function, not a single value. To obtain such a value, defuzzification is needed. In specific cases defuzzification is not necessary because one can already use the fuzzy set as the solution of the problem.

For example, suppose one wants to sell a house. Based on experience, one wants to know what price one can expect. This question is somehow too superficial because the price depends on the time that one considers waiting, i.e., the price to expect in time t . This means the result is a fuzzy function μ depending on time:

$$\mu_{\text{price}}(t) = \text{price to expect in time } t.$$

For CBR this means that fuzzy predicates can constitute the solution of a problem. As another example we mention is the rule introduced in Chap. 9, Adaptation:

IF price is high THEN lower the importance of the attribute price.

This is again a linguistic rule. In order to obtain a quantitative rule one has to formulate this in terms of membership functions and to use an implication as in the Mamdani computation.

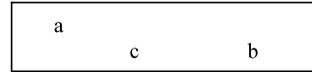
The result of any fuzzy computation is a fuzzy function or a fuzzy function value. In order to perform a certain yes-no decision these values have to be transformed into 0–1 values. This defuzzification process can be done in several ways.

There are four major defuzzification methods which assign to a membership function μ with domain Y an element $m \in Y$.

We put $\text{Max} = \{y \in Y | \forall y' \in Y : \mu(y') \leq \mu(y)\}$ and show the following methods:

- (1) The maximum method. Choose an arbitrary $m \in \text{Max}$.

Fig. 15.5 Spatial relations



(2) Mean-of-Maximum method. Choose m as the mean of Max

$$m = |\text{Max}|^{-1} \sum (y' | y' \in \text{Max}).$$

(3) Centre-of-Area method. Define as the element below the centre of gravity of the area bounded by the curve μ and the y -axis:

$$m = \iint y \times \mu(y) dy Cc \times 1 / \left(\iint (\mu(y) dy) \right).$$

(4) Apply the rough set method. It should be regarded that the first three methods operate using brute force in the sense that they neglect the uncertainty area of the rough set method. In the presence of an uncertainty area one would like to inform the user about it.

Some examples follow:

(1) Spatial reasoning: Spatial descriptions are often presented in a vague form, indicating just the main information instead of giving precise coordinates.

- Fuzzy definitions of spatial areas:

Up right, in the middle, at northwest, at the height of the eyes, ...

- Fuzzy relations between areas A and B :

$$\mu(\text{overlaps}(A, B)) = 0.8.$$

Such fuzzy relations can be applied to crisp as well as to fuzzy areas. Using defuzzification one can again obtain sharp areas from such vague area descriptions again.

The fuzzy interpretation of spatial relations may depend on the intended use as seen in Fig. 15.5. As to the question whether c is lower with respect to a than b is, the answer depends on the influence of the second coordinate.

Such descriptions play a role in image understanding; see Chap. 18, Images.

(2) Customer classes in commerce:

Customer classes are described for treating them properly. They are also market segments and need not be disjoint. Examples are “seniors”, “ecologists”, and so on.

Altogether we have two operations that complement each other:

Fuzzification: Linguistic expressions \rightarrow fuzzy membership functions.

Defuzzification: Fuzzy membership functions \rightarrow crisp function (0–1 decisions).

Table 15.1 Fuzzy classes

Class	Instance
House	House 1
Number of rooms: Integer	Number of rooms: 6
Garage: Boolean	Garage: Yes
Social area: Quality fuzzy	Social area: 0.6
Availability of public transportation: Fuzzy	Availability of public transportation: 0.4

15.4 Fuzzy Sets and CBR

15.4.1 General Relations

CBR and fuzzy rule-based reasoning are two quite successful (approximate) reasoning methodologies. Fuzzy reasoning can be regarded as an integration of logic and approximation. It is done at the cost of the pure logic view somehow no longer being valid: The result of a rule application is no longer a predicate. Similarity reasoning uses approximation concepts to achieve good solutions to problems. The first observation is that similarity can be subsumed under the fuzzy approach. Each similarity measure sim gives rise to a fuzzy membership function μ_{sim} on pairs:

$$\mu_{\text{sim}}(x, y) = \text{sim}(x, y).$$

15.4.2 Fuzzy Cases

Both a case and a query may be fuzzy. A fuzzy case is a case where some of the attribute-values are of fuzzy character. As an example we consider the following object class and an instance. In many practical problems, this occurs regularly. In particular, some attribute-values may be crisp while others can be fuzzy. Table 15.1 shows an example.

15.4.3 Similarities

As mentioned earlier, fuzzy sets can also be the arguments of similarity measures. They extend the application of similarity measures shown so far. This raises the question of how similarities between fuzzy sets should be computed. In order to compare fuzzy membership functions, two methods became popular, which we describe in terms of distances:

(a) Crisp Method:

Select some a_i for which $\mu_i(a_i)$ is maximal, $i = 1, 2$; put $d(\mu_1, \mu_2) := |a_1 - a_2|$

Fig. 15.6 Symmetric difference

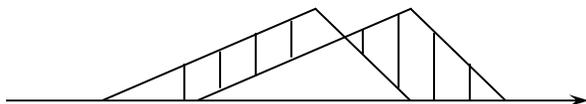
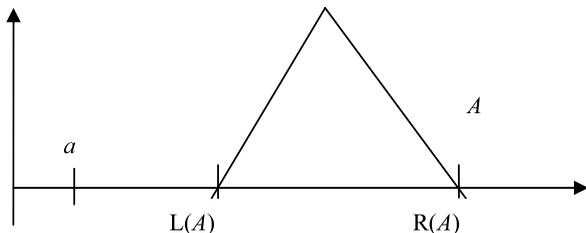


Fig. 15.7 Sets and elements



(b) Integral Method:

$$F_i = \text{Area between } \mu_i \text{ and } x\text{-axis}$$

$$d(\mu_1, \mu_2) = \text{Size}(F_1 \Delta F_2) \text{ (the symmetric difference, shown in Fig. 15.6).}$$

The integral method looks more adequate. It has, however, some problematic aspects that we will see now.

- (a) Two fuzzy functions with disjoint areas have always the same distance.
- (b) The shape of the curves in the crisp method does not play a role.

An improved method is as follows:

- If the areas are not disjoint, apply the integral method;
- If the areas are disjoint, add to the distance obtained by the integral method the distance between the two points where both curves reach zero.

It is also of interest to compare numbers and fuzzy functions, for example, “How similar is 43 years of age to young”? For defining such similarities one should be careful: For $y \in U$ the value $\text{sim}(y, \mu_X)$ should not be the same as $\mu_X(y)$.

For instance, in the fuzzy set high fever, the arguments 37 °C and 36 °C have both membership value 0, but we expect $\text{sim}(36 \text{ °C } (96 \text{ °F}), \text{high fever}) < \text{sim}(37 \text{ °C } (98 \text{ °F}), \text{high fever})$.

In order to compare fuzzy sets and similarity measures we need some simplifying assumptions:

- The universe U is an interval of real numbers (i.e., the approach generalizes to the situation where U is partially ordered).
- We consider n linguistic predicates A_1, \dots, A_n where each predicate A_i has a membership function which attains the maximum value 1 at exactly one argument and is denoted by $m(A_i)$.

Suppose $y \in Y \subseteq \mathbb{R}$ and there is a fuzzy predicate A with membership function μ that has its maximum at $m(A)$; see

Figure 15.7; $L(A)$ and $R(A)$ are, respectively, the left-, and the right-most arguments, where μ has positive values.

Table 15.2 For Temp:
sim_{temp}

	Low	Med	High
Low	1	0.7	0
Med	0.7	1	0.7
High	0	0.7	1

Table 15.3 For Pressure:
sim_{press}

	No	Low	Med	High
No	1	0.7	0.3	0
Low	0.7	1	0.7	0.3
Med	0.3	0.7	1	0.7
High	0	0.3	0.7	1

We assume the compatibility conditions between “<” on the reals and sim:

If $a < b < m(A)$ then $\text{sim}(a, \mu) \leq \text{sim}(b, \mu)$;

If $m(A) < a < b$ then $\text{sim}(b, \mu) \leq \text{sim}(a, \mu)$.

In Figure 15.7, we see: For $a < L(A)$ or $R(A) < a$ the value a decreases monotonically in $|a - L(A)|$ or $|R(A) - a|$, respectively.

More involved measures of this type can occur but they are application-dependent. In particular, one has to look at the utility of the instances of the linguistic predicates because the retrieved nearest neighbour should be the most useful one.

15.4.4 Comparisons

In order to compare fuzzy and CBR methods, we return to the example shown in Fig. 15.3 that shows fuzzy degrees. Now we look at complementary classes {K1, K2}.

First compute the class of a query case using CBR. In the example we assume the following symbolic local similarity measures, as seen in Tables 15.2 and 15.3.

The global measure is obtained by using the following weights:

$$\omega_{\text{Temp}} = 0.7 \quad \text{and} \quad \omega_{\text{Pressure}} = 0.3.$$

The resulting similarity between two points (q_T, q_P) and (c_T, c_P) is obtained by the formula:

$$\text{sim}((q_T, q_P)(c_T, c_P)) = \omega_{\text{temp}} \cdot \text{sim}_{\text{temp}}(q_T, c_T) + \omega_{\text{Pressure}} \cdot \text{sim}_{\text{press}}(q_P, c_P).$$

This measure translates the original linguistic formulations into computable formulas. Suppose we have the three cases C1, C2 and C3 belonging to the classes:

- C1: (high, no): class = K1
- C2: (low, high): class = K1
- C3: (high, high): class = K2

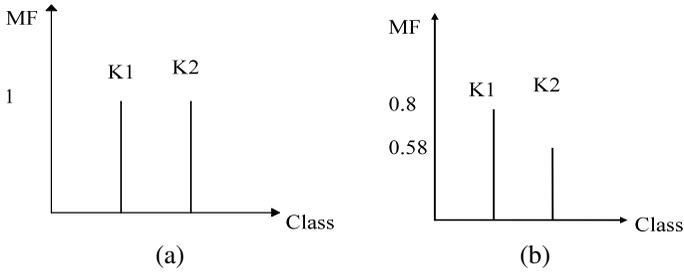


Fig. 15.8 (a) Crisp classes. (b) Fuzzy classification

Next, we take the query $Q = (\text{med}, \text{low})$.

Resulting similarities of the cases for this query are:

$$C1: 0.7 * 0.7 + 0.7 * 0.3 = 0.7$$

$$C2: 0.7 * 0.7 + 0.3 * 0.3 = 0.58$$

$$C3: 0.7 * 0.7 + 0.3 * 0.3 = 0.58$$

Result: The query case is in class K1.

Now we consider this example using fuzzy sets. Attributes become fuzzy (linguistic) variables and concepts that are treated as follows:

- Local similarity measures and related weights become membership functions, one for each fuzzy variable.
- The class also becomes a discrete fuzzy variable (one value for each class).
- Cases are translated into fuzzy rules as follows:
 - Preconditions are *disjunctions* of the form “ A_i is x_i ”.
 - Conclusions are the assignments of the class of the case to the fuzzy variable class using a singleton membership function.

Specific realisations of the fuzzy reasoning approach are:

- Disjunctions in the preconditions of the rules are evaluated using the t-co-norm: $S(a, b) = \min\{a + b, 1\}$.
- The implication is interpreted as Mamdani’s fuzzy implication, i.e., IF A THEN B is realised by

$$\mu(\text{IF } A \text{ THEN } B(x, y)) = \min(\mu A(x), \mu B(y)).$$

- Aggregate the membership functions of all fuzzy rules using
 - $S(a, b) = \max\{a, b\}$ (which is a t-co-norm).
- Take the defuzzification operator Mean-of-Max for achieving a crisp classification.

In the example, we have the linguistic variables X_{temp} and X_{press} . The membership functions are as in Fig. 15.3; a new linguistic variable: X_{Class} for representing the decision class is introduced. It has the values {K1, K2}. The membership function is shown in Fig. 15.8(a) and the classification in Fig. 15.8(b).

The three cases give rise to three rules:

- R1: IF X_{temp} is high OR X_{press} is no THEN X_{Class} is K1
 R2: IF X_{temp} is low OR X_{press} is high THEN X_{Class} is K1
 R3: IF X_{temp} is high OR X_{press} is high THEN X_{Class} is K2

Now we consider the fuzzy reasoning process for the example query

$$Q : X_{\text{temp}} \text{ is med, } X_{\text{press}} \text{ is low.}$$

The fuzzy reasoning proceeds as follows:

Application of rule R1:

- $\max_x \min\{\mu_{\text{temp, high}}^{(x)}, \mu_{\text{temp, med}}^{(x)}\} = 0.49$
- $\max_x \min\{\mu_{\text{press, no}}^{(x)}, \mu_{\text{press, low}}^{(x)}\} = 0.21$

Therefore, we get for the precondition of rule R1 the value $\min\{1, 0.49 + 0.21\} = 0.7$ and for the resulting membership function for the conclusion and the singleton K1 the value is 0.7.

In the same way we obtain the following for the application of R2 and R3: For both, the membership function for K1 has the value 0.58. Aggregating these leads to the fuzzy set in Fig. 15.8(b).

The application of the “Mean-of-Max” defuzzification operator (selecting the value with the maximum membership value) leads to the crisp value K1.

We observe:

- Given one method it is not easy to define the second one precisely.
- The membership functions contain more information than the similarity measure.
- For such problems CBR is more intuitive.

Finally, the question turns up: What can we learn from fuzzy sets for CBR? An interesting point is that we can use different fuzzy membership functions for attribute-values in different cases. Hence it is not necessary to define membership functions globally for an attribute, but enough to do so locally for an attribute and a case. This realises case-specific membership functions.

15.5 Possibility, Necessity, and CBR

15.5.1 General

An important part of fuzzy theory is possibility theory. It deals with “degrees of possibility”. The term “possibility” is hence employed as a graded notion too. A leading idea is to use the notion *possible* as a constraint. It is a kind of weak constraint but in a graded way. In some sense possibility theory deals with the uncertainty area of the rough set method. There is also a strong relation to CBR. In CBR, one says for $\text{sim}(x, y) = u$ that x is the same as y with degree u . Another way to express this would be “the possibility that x is y is u ”. This view will now be explained in more detail.

The basic informational principle underlying the possibility approach to knowledge representation and reasoning is stated as a principle of minimal specificity. This means it excludes only elements where clear restrictions are known. In order to avoid any unjustified conclusions, one should represent a knowledge unit K by the largest possibility measure among those measures compatible with K .

For comparing fuzzy set and possibility measures we consider two example statements:

- (a) Bill has much money: This is a linguistic expression and can be interpreted by fuzzy a membership function μ .

Fuzzy theory asks, for instance, $\mu_{\text{much-money}}(\$2 \text{ million}) = ?$

- (b) Bill owns more than \$2 million: A precise statement.

Possibility theory asks: To what degree do I believe in the possibility that Bill owns more than \$2 million, given that we know that he has much money?

Expressed conditionally: *possibility-degree*($> \$2 \text{ million} | \text{muchmoney}$)?

In possibility theory this degree is defined as $\mu_{\text{much-money}}(\$2 \text{ million})$.

Hence fuzzy sets and possibility distributions use the same mathematical objects but with very different semantics.

15.5.1.1 Formalisms

Possibility measures are defined on subsets of the universe U under consideration.

Definition 15.4 A mapping $\Pi : 2^U \rightarrow [0, 1]$ is a *possibility* measure if

- (i) $\Pi(\emptyset) = 0$;
- (ii) $\Pi(U) = 1$;
- (iii) For all $X, Y \subseteq U : \Pi(X \cup Y) = \max(\Pi(X), \Pi(Y))$.

In a generalization, the values of the measure can be elements of a partially ordered set (P, \leq) . This occurs, for instance, if the real numbers are replaced by qualitative values like {small, medium, high}.

Each knowledge set Σ gives rise to a possibility measure Π_Σ . Depending on the knowledge, certain subsets may get the value 0 because they are (in the naïve sense) impossible. For instance, if Σ says that the deadline is March 31 then the possibility of accepting submissions in April is 0.

Uncertainty of knowledge K can be expressed in terms of a possibility distribution. This is a mapping $\Pi_K : U \rightarrow P$ where U is the universe under consideration.

If there are knowledge sets K and L we denote the conjunction of all elements of K and L by $K \wedge L$. The possibility under this knowledge is computed by the min operator:

$$\Pi_{K \wedge L}(X) = \min(\Pi_K(X), \Pi_L(X)).$$

Now it is of interest that there is a close relationship between possibility and similarity measures. Intuitively, this is a consequence of the following.

For a case (p, s) the similarity of p and p_1 says that s is to some degree also a possible solution of p_1 . Here the measure can be viewed as the quantification of this possibility.

More precisely: Given a problem p and a case (p, s) , according to the possibility measure $\Pi_{\text{sim}}(s) = (\text{sim})$ we can interpret $\text{sim}(p, p')$ as the possibility degree that s is a solution to p too.

Between possibility and certainty, the necessity measures N are located. These measures are defined by:

$$N(X) = 1 - \Pi(U \setminus X)$$

where U is the universe. Again, this is a graded predicate. An event X is necessary insofar as its complement is not possible. It should be remarked that for some X both $N(X)$ and $\Pi(U \setminus X)$ may be nonzero. This parallels to some degree the membership of an object showing it is possible that it belongs both to a set and to its complement.

These measures are strongly connected to the rough set method:

- The elements in the inner approximation are necessarily true.
- The elements in the upper approximation are possibly true.
- The elements outside the upper approximation are necessarily false.

What fuzzy set theory adds here is a quantitative dimension. A way to do it is to define a possibility and/or a necessity measure of the uncertainty region. In Chap. 6, Basic Similarity Topics, we discussed sharp jumps of the similarity measure and suggested introducing an uncertainty region around such a landmark. The semantics of the possibility distribution should reflect the risk of an erroneous solution to the user.

15.6 Tools

A comprehensive tool for rough set analysis is described in Rose2 (2013). There are many tools for fuzzy logic. Sometimes these tools combine fuzzy logic and other techniques like neural nets. As a general reference to this area we refer to www.mathworks.com/products/fuzzylogic/.

15.7 Chapter Summary

The rough set concept provides a method of describing uncertain results in a user-controlled way. Fuzzy sets and their logical descriptions have been introduced. Examples have shown the relations between fuzzy sets and CBR. In addition, possibility and necessity have been introduced. Both have a close relation to CBR. Formalisms are introduced insofar as they are needed for the relations to CBR.

15.8 Background Information

A basic reference for rough set theory is Pawlak (1991). There are many textbooks on fuzzy sets, for instance, Lee (2005) is an introduction. A comprehensive view on the relations between fuzzy sets and possibility theory and CBR is given by Hüllermeier (2007). Extending arguments to allow fuzzy sets are for instance discussed in Dubitzky et al. (1977).

A review of CBR applications that include use of methods for the treatment of uncertainty, though not limited to fuzzy set theory, is given by Cheetham et al. (2005).

15.9 Exercises

Exercise 1 Look for prototypical events for test driving a car. Model this both in CBR and fuzzy set theory.

Exercise 2 Look at your university curriculum in your field. Describe lectures and classes with respect to the utility for you. Model this in both CBR and fuzzy set theory.

References

- Cheetham W, Shiu SCK, Weber RO (2005) Soft case-based reasoning. *Knowl Eng Rev* 20(3):287–304
- Dubitzky W, Schuster A, Hughes JG et al (1977) How similar is very young to 43 years of age? In: *IJCAI 1977: 5th international joint conference on artificial intelligence*, Cambridge, MA, 1977, vol 2. Morgan Kaufmann, San Francisco, p 226
- Hüllermeier E (2007) *Case-based approximate reasoning*. Springer, The Netherlands
- Lee KH (2005) *First course in fuzzy theory and applications*. Springer, Berlin
- Pawlak Z (1991) *Rough sets: theoretical aspects of reasoning about data*. Kluwer, The Netherlands
- Rose2 (2013) ROSE2 <http://idss.cs.put.poznan.pl/site/rose.html>. Accessed 27 Feb 2013

Chapter 16

Probabilities

16.1 About This Chapter

This advanced chapter discusses topics in probabilities of interest to CBR. It is devoted to readers who deal in their applications with stochastic phenomena. Some basic knowledge about probabilities is required.

We consider some basic connections between probabilities and similarity measures. Then some applications to risk analysis are discussed. At the end, stochastic processes and Bayesian methods are considered.

16.2 General Aspects

In Chap. 15, Uncertainty, we discussed uncertainty in general. This chapter is devoted to a special area of this, namely, probability and statistics. Many of the arguments in Chap. 15 apply here too. As an example we mention the use of rough sets. In probability this is an issue too and runs under the term confidence intervals.

In probability theory, similarity has always played an essential role but often with a different terminology. Both views look in many respects into the future. CBR wants to find out what a good solution is. This can depend on the (unknown) outcome of some events. However, some information about probabilities can give a hint about the possible success of a solution. This information can affect the utility of the possible solutions and therefore its similarity measures. We also discuss this in the Chap. 13, Advanced Similarity Topics.

Basically, probabilities can be expressed in two ways, based on frequencies or on subjective opinions. The effect is always that one assigns numbers to events but one interprets them in a different way.

On this level, probability has a theory that CBR cannot challenge. The CBR role is to interact on situations with probabilities. A basic problem for defining similarity measures in this context is that arguments are not fixed magnitudes but frequently random variables. This will lead to some changed view on the properties of the measures.

16.2.1 From Probabilities to Measures

16.2.1.1 Some Measures

The relation between similarity measures and probabilities goes in two directions. On the one hand, probability distributions are used to derive similarity measures. On the other hand, similarity measures can sometimes also be used for getting estimates of probabilities.

In this section we present material that is applicable in frequent situations, i.e., it is of general character. In many situations, the type of similarity is of dynamic character, as introduced in Chap. 6, Basic Similarity Topics, because it compares changing situations. Often, it is the sub type of probabilistic measures. In other sections these methods will be used and extended in different ways.

Probabilities also play a role in the semantics of similarity measures. For instance, in classification one often had the view:

$$\text{sim}(x, y) = \text{Prob}(x \text{ and } y \text{ are in the same class}).$$

This is a quite simple measure and no random variables occur. The involved utility counts how often the class is correct; therefore, the probabilities have to be known.

In Chap. 6, Basic Similarity Topics, we introduced a somewhat more sophisticated measure as the Value Difference Metric (VDM). Such measures are related to the basic meaning of similarity measures, namely, that they reflect utility. If one is not fully informed, one can take the expected utility (as, for instance, in gambling). Then, probabilities are involved but they are combined with other kinds of knowledge that gives biases to the probabilities.

In certain applications, probabilities use the case descriptions directly. For instance, often the problem is to choose an action with the highest expected utility. An immediate idea is to take the probabilities P of events themselves as a similarity measure:

$$\text{sim}_P(A, B) = P(A \wedge B).$$

A refined way uses conditional probabilities, which gives a non-symmetric measure:

$$\text{sim}_{\text{cp}}(A, B) = P(A|B).$$

Besides the fact that only frequencies enter the measure, another disadvantage is that only events and no random variables are considered.

This leads to our taking random variables as arguments of similarity measures. A fundamental difference with the deterministic situation is that one does not simply compare the definitions of the distributions. If in the deterministic case two objects have the same definition then they are identical and it is natural to take their similarity value as 1. Therefore, many similarity measures are reflexive.

In a probabilistic situation one is interested in the distribution of the values of the random variables. Even for one distribution the values will differ from instance to instance. These differences are of interest and therefore similarity measures for

random variables will usually not be reflexive. As we will see, there are different methods to define such similarity measures.

In statistics, similarity measures often run under the term correlation coefficients. We start with some simple measures and arrive at some more sophisticated ones.

Henceforth, random variables are denoted by capital letters X, Y, \dots , and the upper index T denotes the transpose of a matrix. E denotes the expected value and Var the variance.

We consider a vector $(X, Y)^T$ of random variables. A simple and well-known measure is the covariance.

$$\text{Cov}(X, Y) = E((X - EX)(Y - EY)).$$

It measures how much two variables change together. This measure is not reflexive because $\text{Cov}(X, X) = \text{Var}(X)$. The variance is a measure of the amount of variation within the values of that variable.

The next measure is the linear correlation coefficient ρ for $(X, Y)^T$:

$$\rho(X, Y) = \frac{\text{Cov}(X, Y)}{\sqrt{\text{Var}(X) \cdot \text{Var}(Y)}}.$$

Notation: X and Y are

- Uncorrelated if $\rho(X, Y) = 0$.
- Positively correlated if $\rho(X, Y) > 0$.
- Negatively correlated if $\rho(X, Y) < 0$.

This corresponds to positive and negative similarities. ρ is a similarity measure where the values range between -1 and $+1$. It is a stochastic version of the concept of concordance introduced in Chap. 13, Advanced Similarity Topics. The measure can be used to assign weights to the variables: The higher its probability the more important it is. This will be extended below.

Remark for mathematicians: The correlation measures are linear dependencies and this is quite natural for elliptic distributions. For other types of distributions, the situation is, however, different.

Next, probability distributions are compared for dealing with random variables. Two probability distributions P and Q may contain different amounts of information. That means, in order to code samples from P when using a code based on Q , one needs a number of extra bits. In order to measure the difference in the probability distributions, one studies the expected number of these bits. Another way to express this is to say that one wants to know how many unnecessary bits one uses if Q is taken instead of P . Typically, P is based on observations or a given distribution while Q is a model or an approximation. For this purpose a similarity measure is introduced (in the form of a distance).

Definition 16.1 The Kullback–Leibler distance is

$$d_{KB}(P, Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}.$$

This distance is not symmetric and the triangle inequality does not hold. It is also called information divergence or information gain.

The symmetric version is the Jensen–Shannon distance

$$d_J(P, Q) = \sum_x \frac{P(x) \log P(x) + Q(x) \log Q(x)}{2} - \frac{P(x) - Q(x)}{2} \log \left(\frac{P(x) + Q(x)}{2} \right).$$

Both measures will be applied in both textual CBR and images.

16.2.1.2 Risk Analysis

Now we come to risk analysis as an application domain. We consider the situation where one has to select investments from a product base that offers many possibilities. One of the demands is minimal risk. Here risk is defined as the possibility of becoming bankrupt rather than the chance of getting optimal gain. This means here that the risk of the whole value of the investment going down is minimal.

The solution to risk analysis is twofold:

- (1) Which similarities indicate risk?
- (2) When is a situation close to risk?

For this we have to consider a deeper approach and have to look at multivariate (n -dimensional) distribution functions H that are defined for vectors (X_1, \dots, X_n) of random variables from a viewpoint that is of interest in risk analysis. Each X_i represents the probability distribution of one of my investments. We have to find out how variables behave in risky situations. Their probability distributions also follow a *local-global principle*. The central concept for this is the notion of a *copula*. A copula combines local aspects with global ones. In contrast to many other situations in CBR, copulas are of a uniform type and the constructor functions are not completely arbitrary.

Definition 16.2 An n -dimensional copula is a multivariate distribution function $C : (0, 1)^n \rightarrow (0, 1)$ such that:

- (i) if $a_i = 0$ for some i then $C(a) = 0$;
- (ii) if $a_i = 1$ for all $i \neq j$ then $C(a) = a_j$;
- (iii) if $a_i \leq b_i$ for all i then $0 \leq VC((a, b))$ where $VC((a, b))$ is the n th order difference on (a, b) : $VC((a, b)) = \Delta^b a C(t) = \Delta^{b_n} a_n \dots \Delta^{b_1} a_1 C(t)$ and the first-order differences are

$$\begin{aligned} \Delta^{b_j} a_j C(t) &= C(t_1, \dots, t_{j-1}, b_j, t_{j+1}, \dots, t_n) \\ &\quad - C(t_1, \dots, t_{j-1}, a_j, t_{j+1}, \dots, t_n). \end{aligned}$$

In principle, there is a structural analogy to linear weighted measures. Condition (iii) is a monotonicity property.

As mentioned, the relevance of copulas is the fact that they play the role of constructor functions. This is further explained by the following theorem. It shows that there is a uniform way to formulate certain similarity measures.

Theorem (Sklar) *For each n -dimensional distribution function H with margins F_1, \dots, F_n there is a copula C such that for all x one has $H(x_1, \dots, x_n) = C(F_1(x_1), \dots, F_n(x_n))$. Moreover, C is uniquely defined if all F_i are continuous. Conversely, if C is a copula, all F_i are distribution functions and H are defined by the above formula, then H is an n -dimensional distribution function with margins F_1, \dots, F_n .*

This theorem is extraordinary because in CBR there is no general mathematically defined constructor operator that computes all global similarity measures for a class of problems. The linear weighted measures provide uniform representations but in general one cannot define when they apply.

An extension provides a second stochastic version of the concept “having the same monotonicity behaviour”:

Let (x^T, y^T) and $(\underline{x}^T, \underline{y}^T)$ be two observations of the continuous random variables (X^T, Y^T) .

(x^T, y^T) and $(\underline{x}^T, \underline{y}^T)$ are called

- *concordant* if $(x - \underline{x})(y - \underline{y}) > 0$,
- *discordant* if $(x - \underline{x})(y - \underline{y}) < 0$.

This has a probabilistic version where one considers independent vectors (X^T, Y^T) and $(\underline{X}^T, \underline{Y}^T)$ of continuous random variables. Then, between these two vectors,

- the probability of concordance is $\text{Prob}((X - \underline{X})(Y - \underline{Y}) > 0)$,
- the probability of discordance is $\text{Prob}((X - \underline{X})(Y - \underline{Y}) < 0)$,
- the order difference is $Q = \text{Prob}((X - \underline{X})(Y - \underline{Y}) > 0) - \text{Prob}((X - \underline{X})(Y - \underline{Y}) < 0)$.

Concordance has a close connection to the correlation of random variables discussed above. This give rise to define a similarity measure called *measure of concordance* between random variables X and Y . The intention of the concept of a copula is to model dependencies between random variables; therefore it is no surprise that copulas are used for modelling concordance. From the viewpoint of risk, one is, in particular, interested in determining situations where two variables are likely to decrease in value simultaneously. This is quite clear when shares are involved: If the going down of one share results in the going down of all other shares then bankruptcy is not very far. The values of the measure are again taken from the interval $(-1, 1)$.

Let CRV denote the set of continuous random variables.

Definition 16.3 A mapping $\kappa : CRV \times CRV \rightarrow (-1, 1)$ is a *measure of concordance* if

- (1) $\kappa(X, X) = 1, \kappa(X, -X) = -1$
- (2) $\kappa(X, Y) = \kappa(Y, X)$
- (3) If X and Y are independent then $\kappa(X, Y) = 0$
- (4) $\kappa(X, -Y) = \kappa(-X, Y) = -\kappa(X, Y)$
- (5) If $C(X, Y)$ and $\underline{C}(X, Y)$ are the copulas corresponding to (X, Y) and $(\underline{X}, \underline{Y})$ and $C \leq \underline{C}$ then $\kappa(X, Y) \leq \kappa(\underline{X}, \underline{Y})$
- (6) If $\{(X_n, Y_n)\}$ is a sequence with a copula sequence $\{C_n\}$ which converges pointwise to $C(X, Y)$ then $\lim(\kappa(X_n, Y_n)) = \kappa(X, Y)$.

The intentions are obvious: The value 1 means complete concordance and -1 means the opposite; 0 is a neutral value with no dependence.

Condition (5) is a monotonicity property while (6) is a technical requirement. The measure of concordance is a similarity measure (but with range $(-1, 1)$).

A simple example is the measure called *Kendall's tau* (the order difference from above). Let X and Y be standard exponential variables and suppose $(\underline{X}, \underline{Y})$ is an independent copy of (X, Y) . Let $C(u, v)$ and $\underline{C}(u, v)$ be the copulas of (X, Y) and $(\underline{X}, \underline{Y})$, respectively. This means, if F and \overline{G} are the common margins of X and \underline{X} and Y and \underline{Y} respectively, then their joint distributions are $H(x, y) = C(F(x), G(y))$ and $\underline{H}(x, y) = \underline{C}(F(x), G(y))$, respectively.

Next, we define another measure for describing risks:

$$\tau(X, Y) := Q = \text{Prob}((X - \underline{X})(Y - \underline{Y}) > 0) - \text{Prob}((X - \underline{X})(Y - \underline{Y}) < 0).$$

Hence, τ measures simply the difference in the probabilities for concordance and discordance, which is intuitively clear for risk analysis if one assumes that X and Y represent costs.

The measure can be computed according to the following formula:

$$\tau(X, Y) = Q(C, \underline{C}) = 4 \cdot \iint \underline{C}(u, v) dC(u, v),$$

where the integral is taken over $(0, 1)^2$. The factor 4 is due to the fact that the range of the measure is $(-1, 1)$ instead of $(0, 1)$.

In risk analysis the statistical methods based on copulas and measures of concordance play an essential role. This is due to the fact that the risk of bankruptcy increases if a portfolio contains too many similar shares, as was illustrated in the financial crisis in 2009.

However, it was emphasized several times that qualitative, i.e., symbolic attributes, are also important for describing risks. The problem is the lack of integration with numerical attributes in the context of copulas. We suggest here that the similarity-based approach provides a possibility for such an integration method.

If no distribution functions or statistical evidence for the relevancies of the values are available then the risk that several assets of a portfolio drastically go down at the

same time cannot be computed by the measures of concordance that are statistically-based.

Suppose now that there are symbolic attributes describing intuitively some concordance too.

Examples of such attributes are:

- Company type; range = {steel, military, energy, tourism}. For the local similarity sim_{CT} with respect to concordance it is reasonable to assume $\text{sim}_{CT}(\text{steel, tourism}) < \text{sim}_{CT}(\text{steel, military})$.
- Economic structure of the country. If, for instance, the value of one company is likely to decrease on certain events, the value of a similar company in another country may go up or down because of the same events.

Such attributes give rise to a weighted similarity measure where the attributes have to be associated with weights. The local measures and the weights reflect subjective views on domain knowledge, while in the same way distribution functions represent objective statistical knowledge.

Now it is easy to combine such symbolic measures and a measure of concordance into an overall measure: This measure simply has two parts, a numerical part and a symbolic one, that we can weight again.

Next, we show an application situation.

The case base, or, better, the product base, CB is a set of shares; these are intended to be available shares. The problems are sets of shares $S = \{s_1, \dots, s_n\}$. That means a query presents the set S of shares we own and we want to buy a share from CB with minimal risk.

The shares are represented by random variables X_1, \dots, X_n that describe their development.

As a measure we consider first a concordance measure κ . For a problem set S and a share t we define the average concordance between S and t as

$$\kappa_{av}(S, t) = \text{average}(\kappa(s, t) | s \in S).$$

Now we take κ_{av} as a distance measure. This selects the share that is minimally concordant on average to our share, and this is the answer of the CBR system.

This can be varied in various ways. For instance, one can use one of the other measures from above or introduce thresholds.

In Sect. 16.3 on Bayesian reasoning we will encounter other kinds of ways of creating similarity measures from probabilities.

16.2.2 From Similarities to Probabilities

Similarity measures can be used in two ways:

- (a) Estimating probabilities.
- (b) Improving probabilities.

Estimates of probabilities usually arise from counting examples or subjective estimates of experienced experts. But even when the domain can be structured by conditional independence, numerous examples must often be recorded. Without a structure, the source of probability estimates becomes a more pressing issue, especially for agents exposed to novel situations in everyday life. This is where subjective probabilities dominate. Subjective probabilities have much in common with subjective utilities. The difference is that often subjective utilities can be formalized, which is questionable for subjective probabilities.

If we assume the existence of a similarity measure, the situation is slightly improved. If the measure was established on counting examples, there is a natural relation to probabilities. Here, we put ourselves in the position where the source of the measure is unknown. At first glance, it seems to be impossible to derive a probability from such a similarity measure because no connection to a probability can be seen.

For instance, when we are looking for a certain used car and have the demand “colour = white”. In the store there may be just one white car; however, many grey cars have almost the same similarity to the query. Here, the probabilities for a randomly chosen car $\text{Prob}(\text{colour} = \text{white})$ and $\text{Prob}(\text{colour} = \text{grey})$ are very different and cannot be guessed from the similarity.

This shows that without further assumption one cannot draw any conclusions about probabilities from similarities. In specific situations the relation between similarities and probabilities is somewhat closer and this can be used for predicting conditional probabilities.

For this we state some (partially asymptotic) assumptions:

Suppose P is a property and a and c are events where a is fixed and c varies.

- (i) $\text{sim}(x, x) = 1$ and sim is symmetric.
- (ii) If $\text{sim}(a, c) \rightarrow_c 1$ then $\text{Prob}(P(a)|P(c)) \rightarrow 1$.
- (iii) If $\text{sim}(a, c) \rightarrow_c 0$ then $\text{Prob}(P(a)|\psi P(c)) \rightarrow \text{Prob}(P(a))$.

Here (iii) means that $\text{sim}(a, c) \approx 0$ should indicate some kind of independence of a and c .

From these assumptions we can predict conditional probabilities.

Theorem $\text{Prob}(P(a)|P(c)) = \text{Prob}(P(a))^\alpha$ where

$$\alpha = \frac{1 - \text{sim}(a, c)}{1 + \text{sim}(a, c)}^{1 - \text{Prob}(P(c))}.$$

A sketchy proof of this is:

If $\text{sim}(a, c) \rightarrow 1$, then $1 - \text{sim}(a, c)/(1 + \text{sim}(a, c)) \rightarrow 0$; hence, $1 - \text{sim}(a, c)/(1 + \text{sim}(a, c)) \rightarrow 0$, which has $\alpha \rightarrow 0$ as a consequence, and therefore $\text{Prob}(P(a))^\alpha \rightarrow 1$.

A different and simple way is improving probabilities of delivering a correct result several times and by estimating it.

The context is that of making predictions. Suppose there is a case base of past records and a new situation that asks for a prediction. One can estimate probabilities

of the outcomes in general by counting frequencies in the case base. However, this is not a correct estimate for the situation in question because many cases in the case base may have little in common with it.

If now a similarity measure sim is available it can be used to influence the probability by giving more similar cases higher influence than less similar ones. A way to do this is as follows. Cases are of the form (s_i, x_i) where the s_i are the situations and the x_i the predictions. The new problem situation is q and the $p_i = \text{Prob}(x_i)$ are estimated from the case base. Wanted are the probabilities $q(x_i)$ for the predictions of q .

Now we put:

$$q(x) = \frac{\sum_j \text{sim}(q, s_j) \cdot p^j}{\sum_j \text{sim}(q, s_j)}.$$

This gives a more adequate probability. The nearest neighbour search would not do the same. In some sense the procedure can be regarded as a kind of adaptation.

16.3 Bayesian Reasoning

Bayesian reasoning contributes additional knowledge arising from probabilities, which is useful for CBR. It is concerned with the influence relations that we introduced in Chap. 2, Basic CBR Elements. An influence diagram is represented as a graph where the nodes are properties or decisions. An edge from a node A to a node B indicates that A has an influence on B . However, from such a diagram one cannot see how large the influence is. Influences play a central role in linearly weighted measures, where the influence is coded by the weight.

Here starts the contribution of Bayesian reasoning to CBR. It quantifies the influence in terms of probabilities. The way to do this is by annotating the edges by conditional probabilities. This describes stochastic influences. Formally, we get:

Definition 16.4 A Bayesian network is a directed graph.

- (i) The labels on the nodes are probability distributions of random variables.
- (ii) The labels on the edges are conditional probability tables between the probabilities of the two nodes.

Now we restrict ourselves to finitely many events.

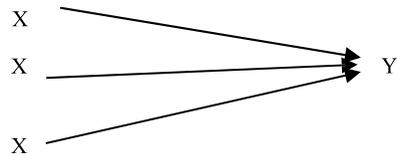
A probability table for two random variables X and Y with the possible events $\{E_1, \dots, E_n\}$ and $\{H_1, \dots, H_m\}$ respectively is of the form in Table 16.1.

The entries are the conditional probabilities $P_{ij} = \text{Prob}(H_j|E_i)$. These annotated networks are called Bayesian Belief Networks (BBNs). In such a network the nodes with no predecessors get their values E explicitly from outside, for instance, by a query or a problem description. The task is now to propagate the probabilities over the net following the path structure. Suppose the variable Y is influenced by m variables X_1, \dots, X_m as in Fig. 16.1.

Table 16.1 Probability table

	H_1	H_2	...	H_m
E_1	P_{11}	P_{21}		P_{1m}
E_2	P_{21}			P_{2m}
...	
E_n	P_{n1}		...	P_{nm}

Fig. 16.1 Influences



We have already computed the probability values E_i for the X_i and are given the possible values $\{H_1, \dots, H_n\}$ for Y . This means the event

$$E = E_1 \wedge \dots \wedge E_m$$

took place. For making the choice of the value for Y , one has the conditional probabilities $\text{Prob}(H_i|E_i)$ from the table. These probabilities are propagated through the whole net along the edges and one gets finally the estimated probabilities at the end nodes (the output nodes). Ultimately, this shows the degree of influence of the inputs on the outputs.

16.3.1 Dynamics

Networks discussed so far have a static character. The dynamic character comes in when a new event takes place that gives a specific value to some variable that was unknown before. This happens frequently when planning and executions are interleaved.

In the belief net this means some node that is not an initial node gets as a value a fact and not only a probability. As a consequence, the probabilities in the net have to be recalculated. Suppose this event is Y in the example in Fig. 16.1.

The task is now to recalculate probability values E_i for the X_i . Unfortunately, one has the probabilities $\text{Prob}(H_i|E_i)$ only. However, what is needed are the conditional probabilities $\text{Prob}(E_i|H_i)$. To close the gap, Bayes' theorem is used.

The general situation is now as follows. Suppose a number of hypotheses H_1, \dots, H_m are given and

- Each hypothesis has a known a priori probability $\text{Prob}(H_i)$.
- For each H_i the conditional probability $\text{Prob}(E|H_i)$ is known.

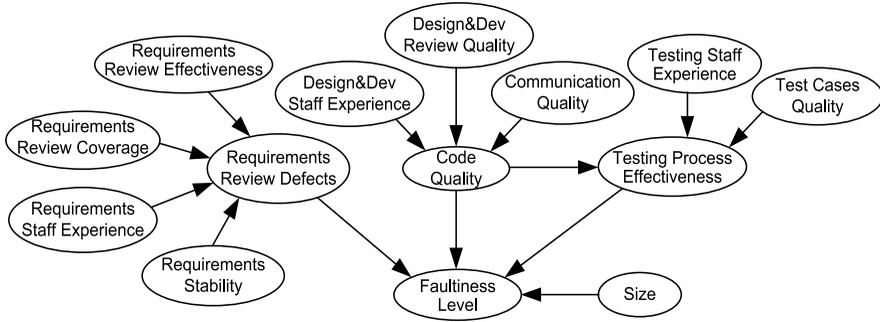


Fig. 16.2 Example Bayesian network

To decide about the choice from among the hypotheses, one takes the one where the a posteriori probability $\text{Prob}(H_i|E)$ after the event E has occurred is maximal (maximum likelihood principle).

This probability can be computed by Bayes' rule:

$$\text{Prob}(H_i|E) = \frac{\text{Prob}(H_i) \text{Prob}(E|H_i)}{\sum_{j=1}^n \text{Prob}(H_j) \text{Prob}(E|H_j)}$$

This rule is basic for all kinds of Bayesian reasoning. It allows for the updating of the net annotations if an event has happened. Inferences in a BBN are calculations of the conditional probabilities along the directed edges. This can be done in a forward or backward mode, where the latter uses Bayes' rule.

Example in a forward mode:

Suppose we have the edge $A \rightarrow B$; then, we get

$$P(B) = P(B|A)P(A) + P(B|\text{not } A)P(\text{not } A), \quad \text{and}$$

$$P(\text{not } B) = P(\text{not } B|A)P(A) + P(\text{not } B|\text{not } A)P(\text{not } A).$$

The influence $P(B|A)$ of A on B can be used in CBR in different ways. A natural task is that of using a BBN for finding the values for the weights in a linear weighted measure. The weights reflect the importance of the attributes, and this can be influenced by different factors that may accumulate in a complex way.

As another example, we consider the problem of predicting the faultiness of software projects. Faultiness is described in terms of several levels, ranging from low to high. The level has to be determined. In Fig. 16.2, the attributes involved are described. The conditional probabilities are omitted because they are updated regularly. There are some virtual attributes but without formal definition. Figure 16.2 shows only what attributes depend on what other attributes; the directed edges show dependencies.

A case is described by the three (virtual) attributes directly below the solution node Faultiness Level. The BBN allows computing the influences (i.e., the weights) of these nodes, starting from the nodes with no predecessor. In these nodes the given data from the query are entered. This means the weights are query-dependent.

16.3.2 Using the Nets

Next, we consider tasks as a whole, for which Bayesian networks can be used to answer probabilistic queries about unobserved variables. From the CBR point of view, the variables are arguments of problem or solution attributes. The observed variables have values from their domain; they correspond to the problem attributes while the unobserved variables have no values and correspond to the solution attributes.

In a probabilistic modelling, one can regard each attribute as a random variable with values from its domain. One can assume that the data vectors are distributed according to some unknown probability distribution. This means BBNs represent general background knowledge rather than specific experiences. However, it should be noted that the Bayesian approach follows directly the CBR techniques by selecting first a solution case and then adapting it.

A Bayesian network can be used to define and compute similarity measures. This is in the spirit mentioned initially for classification: The similarity of two objects is the probability that they are in the same class.

This view is extended now. The network is assumed to connect attributes. If a query is presented, then certain attributes obtain values. The net propagates them in the form of probabilities. Then the attribute-values of the cases obtain estimated probabilities that are interpreted as local similarities. The resulting global similarity defines the nearest neighbour.

16.3.3 Stochastic Processes

We briefly sketched processes and their similarities in Chaps. 5 and 7. Now we extend them to stochastic processes. A stochastic process (or random process) proceeds in discrete steps.

It is defined by a set of random variables $(X_t | t \in T)$, where T denotes discrete times and X_t is the state of the process at time t . The domain of each X_t is denoted by $\{s_1, s_2, \dots, s_t\}$.

Its behaviour is nondeterministic in the sense that the next state of the environment is partially but not fully determined by its previous state; the outcomes of a process step are determined by some probability only.

The likelihood of an outcome at time t being s_j is therefore

$$\text{Prob}(X_t = s_j | X_{t-1} \wedge X_{t-2} \wedge \dots \wedge X_0),$$

i.e., the outcome depends on the history. Classical time series analysis is mainly based on this assumption.

A Markov process has an additional independence assumption that the outcome at time t is independent of all events prior to $t - 1$:

$$\text{Prob}(X_t = s_j | X_{t-1} \wedge X_{t-2} \wedge \dots \wedge X_0) = \text{Prob}(X_t = s_j).$$

This is known as the Markov property. The involved probabilities constitute the process model. If the probabilities are unknown one refers to a Hidden Markov Model (HMM). Similarity queries turn up if one wants to compare stochastic processes.

Examples Garbage collection in cities: Garbage production is a stochastic process and collection creates costs. A case base has stored the process descriptions and the costs in different parts of the city. A query is a process description in a part of the city and the nearest neighbour is supposed to predict a cost estimate. The corresponding probabilities can be obtained from recorded data.

Another problem is when one wants to compare the speed of moving objects in a stochastic way. There are very similar situations. In order to compare two processes, a natural assumption is that they are similar if they have a similar outcome or if one makes similar predictions on their basis. The similarities here are of a dynamic type.

First we introduce some notation. Suppose that two sample sequences $c(k)$ and $d(k)$, $k = 1, 2, 3, \dots, n$, of events are given that are stochastically determined.

Initially we consider just one sequence, $c(k)$. The first step is to determine (or, better, to estimate) the next sequence $c(n + 1)$.

For this we use the linear prediction method. This is heavily used in speech recognition. The basic idea is to predict a value by a linear combination of some (say p) previous values. That means for the sequence c :

$$c(k) \approx \sum_{i=1}^p a_i c(k-i) + u(k).$$

The a_i are the coefficients of the linear function and the $u(k)$ is an external influence factor. The correctness (or a good estimate) depends on the choice of the coefficients a_i , and then the problem is how to determine them. The estimate will contain some error $e(k)$:

$$e(k) = c(k) - \widetilde{c}(k)$$

where

$$\widetilde{c}(k) = \sum_{i=1}^p a_i c(k-i).$$

We collect the errors for all k in some interval I . From this we can estimate the expected value E of error. In order to minimize E we look at the minimum of the expected quadratic error:

$$\min(E(e(k)|k \in I)^2).$$

To compute this minimum the above expression is differentiated with respect to the coefficients a_i and the partial derivatives are equated to 0:

$$\frac{\partial E(e(k)|k \in I)^2}{\partial a_i} = 2E\left(e(k) \frac{\partial E(e(k))}{\partial a_i} \Big|_{k \in I}\right) = 0.$$

This is a system of linear equations that can be solved by classical or more advanced methods. What is obtained is an optimal system of coefficients to estimate the outcome of the stochastic process.

Now we turn to two stochastic processes in the same environment and we compare two sequences c and d . The assumption is that they are similar if their difference $c - d$ is estimated close to 0 on average.

For this reason we apply the above considerations to the difference sequence $c - d$. This can be done in different ways. One possibility is to consider the optimal estimate

$$\sum_{i=1}^p a_i (c(k-i) - d(k-i))$$

for the difference sequence $c - d$. The distance measure

$$d(c, d) = \sqrt{\sum_{i=1}^p a_i^2}$$

measures the distance between the $c - d$ and 0 and hence the distance between c and d .

Again, as with random variables, in general the distances and similarity measures between processes will not be reflexive.

16.4 Tools

The copulas are estimated from observed samples of random vectors. The Matlab (2013) tool calculates the copula, generates random vectors and time series, and provides statistics and diagrams which indicate validity and accuracy of the input model. Matlab (<http://www.mathworks.com/products/matlab/>) also has a tool for linear prediction coding.

There are many tools for Bayesian networks available, for instance: Weka (<http://www.cs.waikato.ac.nz/~ml/weka/>), Matlab, GeNIe (<http://genie.sis.pitt.edu/>), and WinMine Toolkit (<http://research.microsoft.com/~dmax/winmine/tool.doc.htm>).

16.5 Chapter Summary

We discussed that the connections between similarities and probabilities are manifold. The two main directions are:

Probabilities give rise to adequate similarity measures. First we introduced covariance and correlation measures and the Kullback–Leibler measure. Under certain circumstances, probabilities can be estimated from similarities. More sophisticated

measures are measures of concordance. They were discussed from the perspective of risk analysis.

On the other hand, under certain conditions similarity measures can lead to probabilities.

As a way to extend influence diagrams, Bayesian networks were introduced. They contain background knowledge in the form of conditional probabilities.

Process models are often of stochastic character. Linear prediction was introduced, as was a similarity measure.

16.6 Background Information

The background about concordance measures and financial applications is given by Embrechts et al. (2001). Concordance measures play a role in risk analysis. Copulas are a means for a uniform structure for a class of similarity measures in statistics.

Probabilities as discussed are mostly based on models. In everyday life as well as in many applications, this is different. Often, one does not have a model and has not counted frequencies. One rather has subjective opinions about what is more likely to happen. A rigorous discussion of subjective probability is given in Savage (1954) and in Fishburn (1986).

Some information of computing probabilities from similarities is in Blok et al. (2003) and Billot et al. (2005).

For a basic introduction to Bayesian reasoning there are many textbooks, for example, Jensen (1996). More about the combination of Bayesian reasoning and CBR can be found in Aamodt and Langseth (1998).

Stochastic processes have a wide range of applications and developed methods. For an overview, see Brzeźniak and Zastawniak (2000). Here we have shown a very simple approach only. They prefer numerical values while CBR and similarity measures can also deal with symbolic values. For this reason a measure of the HEOM type introduced in Chap. 7, Complex Similarity Topics, is sometimes advisable.

As an introduction to linear prediction, Makhoul (1975) is recommended. The details about stochastic processes and garbage collection are in Niknafs et al. (2011).

Despite the fact that the measures are not reflexive, there is the notion of self-similarity. This concept is used in mathematics in different ways. It mainly says that a *self-similar* object is exactly similar to a part of itself; this is often used for shapes. In the case of stochastic processes it says that a self-similar process behaves the same when viewed at different degrees of magnification, or at different scales in a dimension (space or time). The phenomenon is used in studying networks.

16.7 Exercises

Exercise 1 Consider the development of your favourite sports (e.g., soccer, football, hockey) team. Define a dynamic similarity measure between the teams for predicting which teams score alike. Test this with the results from the previous years.

Exercise 2 Weather is a stochastic process. Define:

- (a) An attribute-value vector for describing weather in your area.
- (b) Weights for the attributes.

Exercise 3 Find areas where subjective probabilities dominate and give reasons for that.

References

- Aamodt A, Langseth H (1998) Integrating Bayesian networks into knowledge-intensive CBR. In: Aha DW, Daniels JJ (eds) Case-based reasoning integrations: papers from the 1998 workshop. Technical report WS-98-15. AAAI Press, Menlo Park, p 1
- Billot A, Gilboa I, Samet D, Schmeidler D (2005) Probabilities as similarity-weighted frequencies. *Econometrica* 73(4):1125–1136
- Blok SV, Medin DL, Osherson D (2003) Probability from similarity. In: AAAI spring symposium on logical formalization of commonsense reasoning. Technical report SS-03-05. AAAI, Palo Alto, p 36
- Brzeźniak Z, Zastawniak T (2000) Basic stochastic processes. Springer undergraduate mathematics series. Springer, London
- Embrechts P, Lindskog F, McNeil A (2001) Modelling dependence with copulas and applications to risk management. In: Rachev ST (ed) Handbook of heavy tailed distributions in finance. Handbooks in finance, vol 8. Elsevier, The Netherlands, pp 329–384
- Fishburn PC (1986) The axioms of subjective probability. *Stat Sci* 1(3):335–345
- Jensen FV (1996) An introduction to Bayesian networks. UCL Press, London
- Makhoul J (1975) Linear prediction: a tutorial review. *Proc IEEE* 63(4):561–580
- Niknafs A, Sun B, Richter MM, Ruhe G (2011) Comparative analysis of three techniques for predictions in time series with repetitive patterns. In: Zhang R, Cordeiro J, Li X et al (eds) ICEIS 2011: 13th international conference on enterprise information systems, Beijing, China, 8–11 June 2011
- Savage JL (1954) Foundations of statistics. Wiley, New York

Part IV

Complex Knowledge Sources

Part IV first deals with case representations that are not primarily based on attributes and their values. These are texts, images, and sensor data. They all can contain knowledge and information and can be used for representing cases. They can also be used for communication and in particular for dialogs. That makes them interesting for conversations.

A basic problem for these representations is to formally understand them. Otherwise one cannot meaningfully fill the other containers and cannot describe queries and answers. A formal understanding is quite different for determining such semantics. For each representation this is described in some detail in the corresponding chapters. They can also be used as queries and the answers can also be of this type. Speech is practical for communication.

These cases can again be stored in a case base. For the other knowledge containers it is, however, not immediately clear what their contents should look like. This is investigated in Chap. 17 for text, in Chap. 18 for images and in Chap. 19 for speech and measured data.

The filling of the containers differs in each case. In the text chapter one makes use of textual and computer linguistic methods. For images one relies on image processing techniques. For speech and measurement data we refer to stochastic processes.

All these problems have in common that we use a version of the local-global view. The local elements are letters, pixels or signals. Going up a hierarchy one eventually leads to the overall description level.

From the CBR perspective we put a particular interest on similarities and retrieval for the different types.

Chapter 20 deals with conversations. These become necessary if the user is not able to formulate a precise query and one wants to make this precise. This requires a dialog between the user and the system. Usually it takes place in one of the media discussed above. In this chapter specific relations between conversations and CBR techniques are pointed out.

Finally, Chap. 21 on Knowledge Management summarizes all the previous chapters. The chapter discusses the numerous affinities and overlapping concepts in CBR and Knowledge Management (KM). We discuss how CBR can be beneficial to KM and illustrate this by explaining applications.

Chapter 17

Textual CBR

17.1 About This Chapter

This advanced chapter is intended for readers who have to deal with knowledge sources in textual format. Readers should be familiar with the contents in Parts I and II. This chapter introduces multiple levels and forms in which textual knowledge is found and used as a source for textual CBR. Understanding of text takes its own form; hence we discuss the problems that text imposes on understanding. This chapter discusses representations that are unique for texts and how they can be used in case-based reasoning. We also describe methods to convert text into the representations that we have studied in previous chapters.

17.2 General

Text is a special way to express knowledge. We refer to text as collections of words in a well-known language that convey a meaning (i.e., ideas) when interpreted in aggregation. These collections are not organised in easy-to-interpret representations such as attribute-value pairs.

Textual CBR is used when knowledge sources for CBR knowledge containers are in textual form. Because text entails knowledge, textual CBR can be defined in terms of the specific form of the CBR knowledge containers. In many areas, textual knowledge has been transformed into some formalism. Textual CBR plays a role when this has not been done, and knowledge to be used in any of the knowledge containers is originally available in textual form. As readers will notice, when knowledge is available in complex representation forms, a specific set of methods is required for their treatment. This is also what happens with images and measurement data.

Queries, problems, and solutions can make use of textual formulations. Because of the lack of explicit formal structures they cannot be understood automatically. When knowledge about problems and solutions is available in textual form, text-tailored methods are needed to produce understanding and effective reasoning.

We are discussing automated methods even though some are considered semi-automated as they require at least one manual step. One characteristic of informal text is that the covered scope of meaning is typically understood by humans because there is an underlying context that is known (rather than artificially formalized). This supports successful communication between humans while creating substantial problems for machines.

There are two main strategies to address textual CBR. One is to convert any text into a machine readable form and manipulate it using the methods discussed in previous chapters. The other is to keep the original textual representation and adapt CBR techniques to manipulate text.

Text is in natural language; it contains its ambiguities, redundancy, and impreciseness that though mundane to humans pose barriers to automatically manipulating text. When aiming to use text for problem solving, text needs to be recognised to the extent that problems and solutions can be identified, compared, adapted, and reused. Sometimes texts are semistructured, as in emails when some machine-distinguishable features such as date and subject are also available. Another variation of semistructured text is when subheadings are given to paragraphs or, in some way, we can determine the topic of a given set of words.

The need to manipulate text overlaps with many other fields such as text mining and information retrieval. There are two views when dealing with texts:

- (1) Document-oriented view: One does not get information about the content of a document but rather about its existence. Documents are referenced by titles or file names only. This view is the one of information retrieval; see Chap. 23, Relations and Comparisons with Other Techniques.
- (2) Content-oriented view: One has access to the content of the document and studies it. A typical document is one that, for example, tells you the difference between object-oriented and functional programming. In this view someone has, at least in principle, to read the whole document to identify its contents. This is the textual CBR view.

The similarity between a new problem and textual cases should indicate the extent to which texts have knowledge of interest to the new problem (because this is its utility).

17.2.1 Text, Structure, and Levels

17.2.1.1 Text

Text is unstructured data that uses words in some language to convey a meaning when their ordering, function, and relationships are known. In order to understand text, first it is necessary to understand the language in which the text is presented. This includes knowledge of the words and grammar. Furthermore, text includes

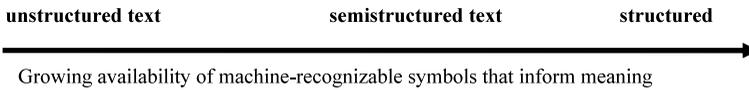


Fig. 17.1 Structuring of text

punctuation and format (e.g., headings), which entail valuable information that can also help convey meaning. The formatting of a text includes empty spaces, and even those may help with understanding. Empty spaces denote the packaging of ideas into words and paragraphs while headings have the power of topic identification and summarization.

17.2.1.2 Text Structure

To better describe the exact challenges textual CBR methods need to overcome, we contextualize text occurrences in a spectrum (Fig. 17.1) moving from unstructured towards structured forms. This spectrum represents the growing availability of machine-recognisable symbols that inform meaning. Note that this spectrum denotes a continuum as perceived by machines.

At the very left is text that has not received any kind of processing; it is merely a collection of alphanumeric symbols like letters that individually make no sense. This is what the machine recognises. An example would be the output of converting spoken language into text.

As we move towards the right, headings may be recognised. Textual content may be packaged within fields of questions and answers as in frequently asked questions (FAQs). These lists typically have headings indicating the beginning of a “question” and an “answer”. The headings for questions and answers do not inform us about the domain but rather indicate a relationship between the contents in them.

More to the right are emails that contain many single-valued features like sender, recipient, urgency, and date. Emails also include the textual feature subject, which may contain from one to many words. The body of the email may be purely textual and thus completely unstructured or a combination of patterns.

At the extreme right of Fig. 17.1 it is no longer text, but a structured form. This is a structure as seen in previous chapters where all the contents are presented in a way a machine can recognise. Dealing with the right-most structure is thus discussed in previous chapters. Here, we are interested in the forms incoming from left to right before the right-most structure.

Before we discuss Textual CBR methods that enable solving problems based on knowledge that is available in textual form, we will further discuss textual forms. This is important because it will help the reader understand the specific problems that semistructured forms can bring. This knowledge is crucial in making choices on which methods to use.

Complexity	Levels	Examples/structure
Simple	Alphanumeric symbols	Keywords, Attributes
	Words	Lexicons
	Relations between words	Sets of words, Thesauri, Glossaries
	Phrases	Sentences, Headings
	Sentences	Question-answers, FAQs
	Hypertext	Cross links between words in sentences
Complex	Free text	Multiple paragraphs

Fig. 17.2 Text levels

17.2.1.3 Text Levels

As previously stated, Textual CBR methods will either convert unstructured text into a structured form or will adapt textual methods to perform CBR tasks. Here we present different text levels. This organisation reflects some form of the local-global principle introduced earlier. The levels are not strictly separated and have access to each other. The organisation of levels is depicted in Fig. 17.2.

The most elementary level is that of **alphanumeric symbols**. These have no meaning unless they are used for denoting words that have a definition.

The components of the level of **words** are built up by alphanumeric symbols (i.e., letters). They have a meaning, and often more than one. The orthography describes the correctness of the spelling of the words. The most prominent uses of words are as keywords or indexing vocabularies. On this level, a dictionary of interesting words is located. A particular kind of word is the well-known attribute, like price or shape. This is the point of departure for acquiring structured knowledge that can be used for generating structured cases.

After words is the level of word collections and **relations between words**. Ordering can be important for word collections. Sometimes, a change in the ordering or a swap of words can cause the meaning to change substantially. Some examples are senior college and college senior, guitar solo and solo guitar. Ordering usually requires text processing, but it is possible to keep ordering of word collections via some representations (see Sect. 17.3.5.3).

There are many kinds of relations between words. They are mostly of semantic nature, and some important ones are:

- a. Synonyms and quasi-synonyms. The latter occur frequently in informal texts and have essentially the same meaning but with minor differences. Such relations are described in a thesaurus. A glossary is application-oriented and contains explanations of special terms. A gloss is a brief summary of a word's meaning, equivalent to the dictionary entry of that word. A glossary is a collection of glosses, as, for example, at the end of a book.

- b. Negations and opposites. Negation is a basic connective in logic. It is the one that constitutes essential difficulties in automatic theorem proving; therefore it is not surprising that it introduces additional problems in texts. Although negation inherits many aspects from logic, in texts negation is not restricted to logical negation. Negation is just one example of a connective. For example, the conjunction “and” is frequently used but not in the logical sense. The concept of “opposite” is very closely related to negation, as in, “Allowed is the opposite of forbidden” and “Cheap is the opposite of expensive”. It can, for instance, include never, the opposite, neither, and so on. For the sentence “The decision is favourable but not unanimous”, all words are required for understanding its meaning. There is a branch of mathematical logic dealing with allowed, namely, deontic logic. It makes these aspects precise. It has applications in areas such as the tax domain.

The level of **phrases** and orderings is the simplest level where words are combined. The combination of words usually has its own meaning. The phrases can be domain-dependent or colloquial in nature. Dealing with them requires understanding. Natural language uses many phrases that combine multiple words to create a meaning. Phrases are not simply sets of words; their ordering is important. Phrases may be recorded in a phrase base.

As discussed in word collections, the sequencing of words can alter their aggregated meaning. These phrases may be separated by multiple words, hyphenated, or simply written contiguously. Depending on the kind of processing, *outcome* and *come out*, for instance, may have the same impact on the chosen representation. Ordering plays a role in the meaning of phrases also at the sentence level because they influence speech roles. In college senior, senior is a noun; in senior college, senior is an adjective.

Ordering may also play a role in the meaning of an entire sentence. For example, “How can I change my schedule from Tuesdays to Fridays”? In this situation, the meaning is preserved only if the correct originating and target days are identified. If the two elements are, for example, routes, the ordering can be even more relevant for the meaning. As an example, consider driving directions in a map, where going from point A to point B can be substantially different than from point B to A.

The most complex level is the one of **sentences**. They combine words and phrases using grammatical rules. The sentences are elements formulated in a language. Grammars (see Sect. 17.3.1.2) describe the correct structure of the sentences in a language. However, understanding their meaning is much more complex.

Common forms occurring at the level of sentences are FAQs. A typical FAQ is of the form in the example shown in Table 17.1, which can be used to produce a solution to an error report. The error report shows that the solution is obtained from the FAQ: “Load and install driver for graphics”.

Hypertext structures are semiformal representations of sentences. They contain formal as well as informal elements. Hypertexts organise (informal) text objects in a (formal) network of nodes connected by edges called hyperlinks. In the next section, we mention how the fact that texts are linear represents a limitation to understanding them. Hypertexts are not linear; they have links that associate different items, creating associative networks. This added dimension enhances meaning, providing

Table 17.1 FAQ

Error report		FAQ	
Hardware:	PC & HP DeskJet 870	Hardware:	IBM Laptop
Operating system:	Windows 95	Operating system:	Windows 97
Error:	My printer crashes for graphic programs	Error:	Certain images are not accepted
Solution:	?	Solution:	Load and install driver for graphics

relationships that facilitate understanding them. Hypertexts are very common on Web pages; a prominent example is Wikipedia. A common example of associated texts is found in email threads. There are many other semiformal representations of text, such as word vectors and templates, but these are discussed in sections where we present methods to construct these representations as a means to add structure to text (e.g., Sects. 17.3 and 17.4).

The **free text** is the last construction for presenting sentences. Usually, understanding requires knowledge that is not explicitly mentioned. That makes automated understanding difficult. For this reason we will discuss various methods in later sections.

As we move from left to right, there are two groups, the problems we face in understanding texts and the methods for overcoming them. Before we move into the main sections about methods, we first discuss properties that may create or solve problems when manipulating texts.

17.2.2 Text Properties

This section extends the discussion of the characteristics of text. Here we focus on those characteristics that create or resolve problems when reasoning with text. Identifying the presence of some of those characteristics and consequently the potential problems or benefits that may arise when using a given text for reasoning can be useful in selecting the most suitable textual CBR method.

The properties described in this section occur in texts that are typically used in Textual CBR applications, for example, emails or documents used by organisations containing know-how. The presentation provided here is thus limited to problems frequent in typical domains. This excludes less usual problems like the ones that occur in literary texts such as those of Shakespeare. These are some properties to consider:

- Reliability and provenance,
- Text segmentation,
- Linguistics,
- Limited themes.

Having further information about the author of a text or the text **provenance** becomes relevant particularly if the goal is to identify a suitable technique to automate text representation, retrieval, and reuse. Knowing the original purpose of a given text collection, as well as information about its authors, is crucial in helping us decide how to process it. Texts are not written with the goal of being used in Textual CBR. If that were the case, then structured cases would be directly constructed. Therefore, knowing the original purpose, the homogeneity among authors and their ability as writers are all important matters.

Reliability is also related to provenance, but now we emphasize the continuity of reliable text sources. The frequency with which texts become available and how constant their characteristics (e.g., correctness) are have to be analysed. It is a constant evaluation of all the above aspects across the dimension of time.

One element of style is the separation of **text segments** through paragraphs. Paragraphs should be designed with topical and supporting sentences in support of a message around one theme. Stylistic guidelines recommend creation of new paragraphs once an idea is completed and another one starts. Hence, as long as a given text has more than one paragraph, it is reasonable to assume that it can be broken down into smaller segments. The automation of text segmentation is typically done by seeking topical boundaries. This is typically accomplished with long range n-grams, linguistic patterns, and trigger pairs. More on this where we discuss how to create hypertexts (Sect. 17.4.1).

Linguistics offers theories of rhetorical structure, which identify functions or purposes of portions of text. The function structure may not perfectly map to paragraphs because more than one paragraph can support the same function. It is more likely that one function will be present in more than one paragraph. The theory of speech acts provides further insights into the functions of text excerpts. For example, commitments and the impact caused by text (e.g., we acquit) are some of the functions that can be identified. A large number of texts should be available where similar structures recur, so it is worthwhile designing automated methods to recognise these structures.

An example is a case base for document drafting. Organisations typically produce documents that have the same basic ideas; their creation is a design task. Reusing and adapting previous versions via CBR requires documents to be index-based on the purpose or function of each excerpt. Cases are represented through functional structures, each characterised by speech acts, which can be detected by a combination of verbs and nouns that characterise, for example, intent, as in, “We will deliver...”.

The fact that **themes are limited** is a positive thing for text processing. Texts with limited themes may or may not be domain-specific; they may comprise a few different domains, but they are not open-ended like literary texts. Themes have specific expressions with specific meanings, making it easier to define and structure the vocabulary.

These properties characterise texts, causing the automation of problem solving to be sometimes challenging. Typical situations are those where knowledge that could be used to help in problem solving is hidden in textual documents such as legal cases

that are recorded in text all over the world. They embed how problems are solved and how each legal system provides an outcome to a legal problem. Technical faults when frequent are typically recorded in organisations. The general lessons learned are collected and stored in textual form in all types of organisations throughout the world. Informal and infrequent problem solving these days is usually documented in emails, even if unintentionally. The technological context we live in today is likely to have enough knowledge captured in text for significantly advancing any field of science.

We now discussed general properties. Next, we look at specific problems hindering the use of texts for reasoning.

17.2.3 Problems in Understanding Text

The following are problems that textual forms entail that make it hard to understand text. As we discuss some problems, we mention some methods that may be used. This section is later complemented by methods organised by container: vocabulary, case base, similarity, and adaptation.

17.2.3.1 Unidimensional Structure

Unstructured textual documents are characterised by a one-dimensional structure. A simple example is a paper dictionary where every word has one definition whose words are not associated with anything else. Thus, if you look up the term piledated in a paper dictionary (or a bad electronic one), you will find a definition such as, “having a pileum”, although you obviously do not know what a pileum is or otherwise you would have guessed what the adjective “piledated” refers to. A nonlinear structure would have the word pileum in the definition linked to its own definition (i.e., a crest).

The way to deal with the linearity of text is to identify words that are significant in the overall understanding of a text excerpt and adopt a representation that circumvents this limitation. The solutions vary from creating hypertext structures (Sect. 17.4.1) to creating representations that associate words that co-occur in similar contexts (as in latent semantic indexing in Sect. 17.3.5.1), or yet to explicitly adding meaning (as in information entities Sect. 17.4.3).

17.2.3.2 Language

Language comprises expressions chosen by an author, which can create different contexts for text understanding. Some texts make an extensive use of proper nouns, for example, descriptions of personal events, where people are typically referred to by their names, as in “Alex, who is 14, loves reading”! But proper nouns do not make good indexing words; they do not recur as when individual roles are used.

This problem can be dealt with by identifying the roles of each proper noun and replacing them with their roles before they are subject to indexing. The previous sentence could read, “My brother, who is 14, loves reading”!

Titles used in social conventions are also helpful for indexing. Terms associated with forms of treatment such as counsellor, senator, professor, your honour, and so on, can be used to identify roles.

Languages can vary in multiple dimensions. The language can vary too (e.g., Nepalese, Spanish) and they can vary, among others, in the terminology used (e.g., car repair, legal). When the language is domain-specific, then domain-specific glossaries and thesauri can be used to increase the generality of words for indexing. Something as simple as two car models can be easily generalized by the use of a domain-specific thesaurus. For example, an Audi Q7 is a sports utility vehicle and a BMW 328 is a car.

17.2.3.3 Polysemy and Polymorphism

The rich and not well documented vocabulary of natural language has words and expressions that are not uniquely defined and may not be context-dependent. This raises the need to find the meaning of expressions.

The coexistence of many possible meanings for a single word is known as polysemy. This causes a phenomenon known as semantic ambiguity—when one word has multiple interpretations, causing its interpretation to be ambiguous. This phenomenon is sometimes referred to as homonymy, indicating that there is more than one way to describe or interpret the same word. For textual CBR, clarification of polysemous words should be done before texts are processed or used for reasoning. For example, the word crane in a collection of documents about litigation could refer to a proper name, to a construction machine, or to a skull.

Polymorphism is when a concept can be represented through multiple forms. This can be observed with the existence of synonyms, for example, in nouns like clothing, wear, outfit, garment, livery, costume, attire, and so on. Another type of polymorphism in natural language has been identified as words that are similar in their impact, role, or ability to characterise an excerpt in a given context. These groups of similar words are used only after a context has been identified. These words can be, but are not necessarily, synonyms. For example, in the context of elective plastic surgery, words such as complications, death, coma, and agonizing may all refer to undesired consequences. Polymorphic words are important when defining the vocabulary.

The following sections describe methods organised by container: vocabulary, case base, similarity, and adaptation.

17.3 The Vocabulary Container

In ordinary language, vocabulary means the set of words one knows or uses whose meanings are available. In textual CBR, the vocabulary container determines the set

of words used in textual queries and solutions in the representation that is used for reasoning. To be used for reasoning, some structure is needed. Most methods we discuss in this chapter attempt to use representations to give text some structure. Recall Fig. 17.1, where text on the left has no structure. This is where we start this section. Each method is an attempt to move the original text from left to right on the scale of structure. We start with some basic notions of syntactic parsing, n-grams, bag of words, and vector representations; all can be used to represent cases.

17.3.1 Text Processing

The main task in text processing is syntactic parsing, which converts text into a parse tree. A variety of preprocessing steps can be used to facilitate syntactic parsing. A collection of parse trees can be used for case representation.

17.3.1.1 Preprocessing

Preprocessing may include steps where text is spell-checked, and stemmed, apostrophes are interpreted, acronyms are spelled out, proper names are replaced by their roles, stop words are removed, and so on.

The commonly used stemming is the process of replacing words with their stems. A stem of a word is that part which it has in common with all its inflected variants. Removal of stop words is a different process that has the risk of weakening understanding while reducing dimensionality. A stop word is a text filler that can be removed without creating a substantial reduction of information and meaning. The removal of stop words generates a bag of words (see Sect. 17.3.3). The resulting representation is limited but keeps the meaning of the main components such as verbs, nouns, and adjectives. Examples of stop words are articles, (e.g., a, the) and prepositions (e.g., of, to). The purpose is to simplify the text without losing information which is often contained in free texts.

It is important that the nature of the texts be examined so the need for preprocessing steps is understood. Replacing proper names by their roles can significantly improve similarity accuracy, for example. Removal of stop words may hinder syntactic parsing; this can be understood next.

17.3.1.2 Syntactic Parsing

The traditional method to process an unstructured text sequence in natural language is syntactic parsing, whose result is a syntactic structure—typically, parse tree. This general concept is laid out in Fig. 17.3.

Parse trees (right side of Fig. 17.3) are graph-oriented structures that represent the arrangement of words by assigning syntactic roles through the use of context-free

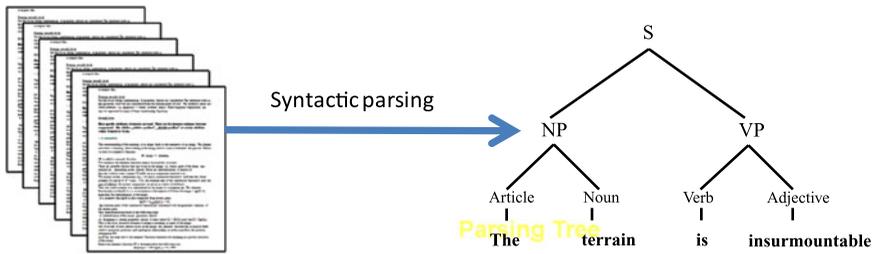


Fig. 17.3 Syntactic parsing

$S \rightarrow NP VP$ $S \rightarrow VP VP$ $S \rightarrow VP PP$
 S = subject; NP= noun phrase; VP=verb phrase; PP=prepositional phrase.

Fig. 17.4 Example of a grammar

grammars. A grammar is a body of knowledge that dictates how syntactic roles are organised. For simplicity, let us see grammars as consisting of rules that define how words and sets of words can be presented in a language. For example, in English, one says, “Warner loves Elizabeth” rather than “Warner Elizabeth loves”. The way a grammar would dictate this truth is by stating that there is a rule in English that has an organisation noun, verb, noun and not a rule stating an organisation noun, noun, and verb. Of course, the rules are not this open ended, they are better represented, and they organise phrases by grouping words and other word sets such as noun phrases and verb phrases. The actual representation of the rule for the example is

$$S \rightarrow NP VP,$$

which means that a sentence can be composed of a noun phrase followed by a verb phrase. An example of a grammar is in Fig. 17.4. Note that a complete grammar also includes symbols like the arrow, which indicates that elements on its left are different from elements on its right.

Identifying the functions of words helps understanding and adds knowledge and structure, helping further manipulation for reasoning. One of the uses of parse trees is as an intermediary representation to be used in deep language processing.

There are basically two main types of parsing algorithms to use grammars to convert text into parse trees.

- (1) The first type implements parsing as a search problem, varying it with dynamic programming to deal with ambiguity.
- (2) Statistical parsing assigns probabilities to a syntactic interpretation of a word or string and uses it to select the most probable one. This can be done, for example, through training with a specific corpus where the probabilities are captured.

17.3.2 N-Grams

N-grams are representations built from text. N-grams are sequences of length n . This length may be regarded as the number of letters or words. Here, we exemplify n-grams using letters. Using the standard method, all the 2-gram (i.e., bi-grams) from the sentence “I was no shadow of a king” becomes:

IW, WA, AS, SN, NO, OS, SH, HA, AD, DO, OW, WO, OF, FA, AK, KI, IN, NG

Analogously, the approach where all possible three-letter combinations of a word are considered is called trigrams. Examples follow.

ANTHRAX: ANT, NTH, THR, HRA, RAX
 ANTHRACIS: ANT, NTH, THR, HRA, RAC, ACI, CIS

There are also variants like the gapped n-gram. The gap “_” represents a character that is not to be used for similarity purposes.

ANTHRAX: (AN_H, NT_R, HR_X)

The benefit of n-grams to CBR is that they can be used as an intermediary case representation form. Some n-gram forms play a role similar to stemmers in that they represent the words with different prefixes and suffixes with common n-grams, allowing their similarity to be revealed when n-grams are compared. See the example of anthrax and anthracis. Methods such as space padding are also helpful when dealing with typos.

17.3.3 Bag of Words

Bag of words (BOW) is the simplest representation form where a set of words is tokenized without keeping their original ordering or any syntactic information. The BOW is an alphabetically ordered comma separated list of the words from the target sentence or excerpt. This can be implemented with or without preprocessing steps such as those of removing stop words or using a stemmer.

BOW is a set with n tokens $t \in T$:

$$T = \{t_1, t_2, \dots, t_n\}.$$

Given the sentence, “I hate Mondays and love Fridays”, the BOW is:

$$T = \{\text{and, Fridays, hate, I, love, Mondays}\}.$$

The example illustrates the impact on understanding as it is no longer identifiable the day that is loved or hated. The loss of ordering can influence the understanding of the original meaning. Bag-of-words are frequently used for case representation. This should be used with caution.

17.3.4 *Vector Representations*

Consider the following text with 30 unique words: “It was merely a great large village; and mainly mud and thatch. The streets were muddy, crooked, unpaved. The populace was an ever flocking and drifting swarm of rags, and splendours, of nodding plumes and shining armour”. The word vector representing this text is a 30-dimensional vector valued with frequencies of the words in the text. At the position of word “and”, its value is 5; for the positions of “of”, “the”, and “was”, the values are 2. All other positions are valued 1.

By removing stop words, the vector representation would change and all positions would have value 1. Once a stemmer is used, the position for mud would receive value 2, and all others 1. Regardless of how a vector is built, the value represents the relevance of a word; thus we refer to it as a weight. Understanding how these representations are created allows us to know how they impact words that hold relevant meaning for indexing, reuse, and adaptation.

The resulting space of words can be used to represent documents as well. Some models choose to create the space from documents and then represent words in the space of documents.

Let the number of selected terms be m ; then an m -dimensional vector is generated that has for each term some associated weight.

A document vector is of the form

$$d = (w_{1j}, w_{2j}, \dots, w_{mj})$$

and a query has the format

$$q = (w_{1i}, w_{2i}, \dots, w_{mi}).$$

Vector representations are very useful in textual CBR; they are standard in information retrieval; see Chap. 23, Relations and Comparisons with Other Techniques.

17.3.5 *Distributed and Reduced Representations*

Thus far we have discussed representations for text that can be categorized as localist. Here we introduce distributed representations, which are very useful for textual CBR and also relevant for images. Next, we introduce the overall concept of distributed representations and follow by describing its main instances, namely, latent semantic indexing, random indexing, and holographic reduced representations.

Distributed representations use interconnected units to describe concepts. Local representations require one unit per concept, whereas distributed representations can describe multiple concepts with one unit. While one concept can be represented with

multiple units, the same unit can participate in the representation of multiple concepts. Distributed representations create a continuous and high-dimensional space where units and concepts are described.

Distributed representations can have problems. Sometimes, multiple elements can be misinterpreted after superimpositions. While several objects can be represented and superimposed, their relationships will not be described. Distributed representations are not well suited to represent predicates, and translate concepts into a localist representation. Distributed representations cannot easily define the degree to which an object is described.

The benefit of these representations for textual CBR is in the embedding of higher-order relations. This makes a difference when assessing similarity between textual cases compared to simpler forms like bag of words, which do not embed higher-order relations. Consequently, with distributed representations, a new litigation case that involves an accident caused by a crane would retrieve litigation cases with accidents related to construction equipment with a higher similarity score than cases with accidents involving Mr. Crane.

Reduced descriptions increase the ability to represent complex structures in distributed representations. The method to create reduced descriptions should:

- Allow the reconstruction of the full representation from the reduced form.
- The reduced form should use fewer units than the full form.
- The reduced form should be related to the full form systematically.
- The reduced form should be informative, so that while a reconstruction becomes necessary only when the reduced object becomes the focus, it is still able to help describe other objects.

Before we discuss examples of distributed and reduced representations used to model words in a space, we discuss the concept of context vectors. Context vectors or semantic vectors are vectors that entail contexts of words or documents. There are two basic ways to build, characterise or limit the context of words:

- (1) Other words occurring in a limited surrounding region or window;
- (2) Other words occurring in the same document, which is typically defined by the application.

There are also two basic ways to build word context vectors:

- (1) One row for each word and one column for each document;
- (2) One row for each word and one column for each word in the entire vocabulary.

17.3.5.1 Latent Semantic Indexing

Latent Semantic Indexing (LSI) is an indexing method based on the assumption that a text is a random choice of words to describe a latent semantic structure. For example, when we say, “men and children”, the choice of words “men” and “children” was a random act of the author, who could have very well chosen other words. This

implies that the sentence entails a semantic structure that can be represented in multiple ways, such as “humans and their offspring”, or “experienced and inexperienced people”, or “adults and minors”, and so on.

The purpose of LSI is to attempt to identify such a semantic structure and use it for indexing and retrieval purposes. Hence, it is a form of text representation that attempts to enhance the limited meaning embedded in words. It is important to note that LSI succeeds in representing the semantic structure of words by providing a representation that includes other words that have associated meanings. For this reason, it can overcome limitations originating from polysemies, but their method assumes one meaning for each word. It is unable to account for ambiguous words that have several different meanings.

As in the example given earlier of litigation cases, for textual collections that require a fully automated method, LSI can help identify documents that are similar to each other. The higher-order relation embedded in the representation precludes the manual or supervised definition of the vocabulary container. If more detailed reasoning is required, LSI can be used to indicate categories of cases to be further studied.

The approach creates a space that associates terms with documents, using singular value decomposition to help organise the space by prioritizing stronger associations from the data. The interpretation is that strong associations recognise patterns that occur in the data. This space then becomes a map of associations of terms from the data, so that a position in this space represents a form of indexing. Hence, words that were used in a query help identify a point in the space where the nearby documents are to be retrieved.

Let A be a rectangular word document matrix, $m \times n$:

$i = 1, \dots, m$, the number of terms, rows in A

$j = 1, \dots, n$, the number of documents, columns in A

The singular value decomposition (SVD) of A is given by:

$$A = USV^T.$$

A , also referred to as semantic or context vector, is decomposed into the product of the three matrices U , S , and V^T such that U and V are orthonormal and S is diagonal; V^T is the transpose of V . The diagonal elements of S are all positive and ordered in decreasing magnitude.

As with other representations, SVD decomposes a matrix to generate a representation with reduced dimensionality. As it does this, it compresses the representation, causing concepts that share a substructure to become closer. Hence, documents that share similar associations tend to get closer in the space.

Therefore, a query containing the term *men* will retrieve documents associated with terms *humans* and *adults* as long as the data collection reveals strong associations between those words. If the associations are weak or inexistent then they would not appear in the semantic space in the same region.

The two orthonormal matrices U and V represent, respectively, words and documents. Similarity between words is given by the dot product or cosine between

vectors representing two objects. Further details on using the cosine for similarity are presented in Sect. 17.5.4.1.

The dimensionality reduction of SVD and similar methods such as principal component analysis also presents drawbacks. They are not incremental. Every time new documents are added, the process has to be repeated. This is very expensive in terms of both memory and time. The literature suggests methods that avoid the initial cumbersome matrix, such as random indexing, discussed next, and possibly adopt a method that may be able to provide some intermediary results as well.

17.3.5.2 Random Indexing

Random indexing (RI) is an incremental distributed representation. The basic idea of random indexing is to randomly select a small subset of vectors to represent an object. As will be explained, typically, random indexing is used to represent documents from a collection of dimension n . Instead of using n vectors, one for each document, a fixed and much smaller number of vectors is defined. This is a number that the designer chooses to represent a more tractable number of dimensions, say n' . Some examples in the literature use n' ten to 20 times smaller than n . The idea is to have all n documents in a distributed representation. Another number is set by designers, the number of vectors to be used to represent each document, n'' vectors. The result is that instead of representing word-document frequencies in a matrix A , $m \times n$, they are represented in a matrix B $m \times n'$, where each document occupies n'' vectors. Note that one same vector is used to represent multiple documents. As for the value of n'' , it can be any number such as 15 or 300. This choice has obvious consequences. Each document is then represented by a document's random index vector. Such a random index vector is the n' vector with values, typically 1s, populated on the n'' vectors or columns corresponding to each document. This association is referred to as an activation of those vectors or columns, that is, each document that was assigned a set of columns activates those columns. Variants using probabilistic distributions for assigning 1s and -1 s are also used.

A further decomposition step like SVD in LSI may be also used. It is also possible to create random index vectors for words. Once the dimensionality is substantially reduced to one of the objects, the other may simply be left with the localist representation. However, if the vocabulary changes and new words are included, then using random index vectors for words allows the originally chosen dimensionality to remain the same.

Besides the reduced dimensionality, RI can also populate the entire matrix at once, allowing intermediary results. Because this dimensionality is determined ahead of time, the matrix does not have to change even when new data is added to the set.

What makes RI possible is the notion that orthogonality can be approximated. It is the Johnson–Lindenstrauss lemma that guarantees this. The lemma states that a set of n points in a high-dimensional Euclidean space, can be mapped into a randomly

selected $O(\log n/e^2)$ -dimensional Euclidean space resulting the difference between any of two n points changing only by $(1 + -e)$.

RI is thus recommended for larger collections that would make other methods too computationally expensive. The fact that it is incremental would also make it amenable to large collections of documents that are being constantly populated, such as lessons learned and legal cases. Once RI is implemented, the resulting representations can be used for similarity assessment using text-to-text measures such as the cosine (see Sect. 17.5.4.1).

17.3.5.3 Holographic Reduced Representations

Holographic Reduced Representations (HRRs) are distributed representations and are also reduced descriptions. The algebra involved in HRRs is beyond the scope of this book; hence we limit the discussion to when and why this representation can be a choice.

In the previous section, we explained how distributed representations are handled in RI. In HRR, the process of creating the context vectors is equivalent to that in RI. What distinguishes HRR is the use of circular convolution. Convolution is a commutative multiplication operation between vectors that produces a resulting vector that can be seen as a compression of the two convolved vectors.

The word space models we have discussed thus far are limited to represent words; no grammar is included. They are advantageous in that they represent higher-order relations, which combined with the cosine measure for similarity can find semantic similarity between texts without any preprocessing. Furthermore, multiple ways of representing word order can be applied in conjunction with distributed representations. The general idea is to take the phrases whose sequences need to be preserved and maintain their representation as sequences. Section 17.10 (Background Information) includes references to how this can be done.

17.3.6 Other Representations

The universe of text representations is vast. There is a wealth of representations that have been proposed in other disciplines, such as information retrieval, that are not discussed here though they are potentially useful for textual CBR. Information retrieval, databases, and pattern recognition are discussed in Chap. 23, Relations and Comparisons with Other Techniques.

17.3.7 Identifying and Enhancing the Vocabulary

As we continue to consider how to move texts from left to right in Fig. 17.1, next, we discuss methods used to identify and enhance the terms the system uses for un-

derstanding and manipulating problems and solutions. These methods do not necessarily involve representations; so they can be potentially used with any of the previously discussed representations.

In the next sections we present feature extraction; then we list external sources that can be used for enhancing the vocabulary to increase meaning. We conclude by presenting word sense disambiguation, an example method that uses external sources to treat a common source of textual uncertainty.

17.3.7.1 Feature Extraction

Feature extraction is a learning method that aims at capturing a set of features that is sufficiently representative such that it works as an effective model of the original. Feature extraction is also a technique for reducing dimensionality, making it desirable when knowledge sources are available in textual format, as well as in images or speech (see Chaps. 18 and 19).

The goal of feature extraction in CBR is to identify indices for guiding retrieval. Therefore, the same characteristics we expect of a good index can be used as reference criteria for feature extraction.

One of the strategies used in feature extraction is the use of information gain as a criterion to identify discriminatory features. This has been mentioned in Chap. 4, Application Examples, and is further discussed in Chap. 22, Basic Formal Definitions and Methods.

17.3.7.2 External Sources

In order to enhance vocabulary, external sources are required. Two types of auxiliary sources are described next, with two ways to proceed:

- (1) Sources like dictionaries, thesauri, or ontologies. These sources are mostly human-made and generally built to be used by humans; they are also well structured. Ontologies offer two major sources of information on words beyond just a set of words; they give rise to extended structures:
 - a. The ordering structure of the ontology.
 - b. The frequency with which words occur in parts of the ontology.
- (2) Arbitrary texts. They are also mostly human-made but their original purpose is not the use we discuss here. These are quite large text corpora that represent through many examples how humans use the language.

Thesauri include synonyms, quasi-synonyms, and possibly properties of words and relations between them, such as:

- Semantic relations between words,
- Different meanings of words,
- Classification with respect to syntactic categories,

The screenshot shows the WordNet Search interface. At the top, it says "WordNet Search - 3.1" with links to the home page, glossary, and help. Below this is a search bar with "crane" entered and a "Search WordNet" button. There are also "Display Options" and a "Change" button. A key explains that "S:" shows synsets and "W:" shows word relations. The results are organized by part of speech: Noun and Verb. Under Noun, there are five entries: Crane (writer), Crane (poet), Crans (constellation), crane (machine), and crane (bird). Under Verb, there is one entry: crane (stretch).

WordNet Search - 3.1
 - [WordNet home page](#) - [Glossary](#) - [Help](#)

Word to search for:

Display Options: (Select option to change)

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations
 Display options for sense: (gloss) "an example sentence"

Noun

- **S: (n) Crane**, [Stephen Crane](#) (United States writer (1871-1900))
- **S: (n) Crane**, [Hart Crane](#), [Harold Hart Crane](#) (United States poet (1899-1932))
- **S: (n) Crans**, **Crans** (a small constellation in the southern hemisphere near Phoenix)
- **S: (n) crane** (lifts and moves heavy objects; lifting tackle is suspended from a pivoted boom that rotates around a vertical axis)
- **S: (n) crane** (large long-necked wading bird of marshes and plains in many parts of the world)

Verb

- **S: (v) crane**, [stretch out](#) (stretch (the neck) so as to see better) "*The women craned their necks to see the President drive by*"

Fig. 17.5 Results from searching word “crane” in WordNet

- Explanations of meanings of words,
- Other semantic relations.

An example of a thesaurus, also called a linguistic ontology or lexical database, is the frequently used system WordNet. It is an annotated graph where the nodes represent real-world concepts. The annotations on the edges represent relationships between these concepts. For instance, “car *is a kind of* vehicle”, “high *is opposite of* low”, and so on. See Sect. 17.5.2 for similarity measures that make use of WordNet.

WordNet knows four parts of speech: nouns, verbs, adjectives, and adverbs. It contains word forms, word meanings, types of relationships, and noun and verb *is-a* hierarchies. The representation is only semiformal. This means that the meaning is not fully machine understandable. An example of the information available in WordNet is shown in Fig. 17.5.

Other ways to use WordNet include assessing similarity between words. Section 17.5.2 discusses structure-oriented similarity measures. Using those functions in a structure like WordNet allows the automated comparison of terms for similarity assessment. Another use is for word sense disambiguation, discussed next.

17.3.7.3 Word Sense Disambiguation

Another way to go from left to right in Fig. 17.1 is disambiguation. It adds more structure as it enhances meaning towards a better understanding. Disambiguation is an isolated step to resolve polysemous words. The solution to semantic ambiguity is

Table 17.2 Capturing senses of a target word

Target word	Noun Sense 1	Noun Sense 2	Noun Sense 3
Bank	raised shelf ridge ground	long seat several sit bench form platform stage speak	money dealer table counter shop

Table 17.3 Capturing senses of words in a window

Sense 1	
Deposit	Sum money deposited bank usually interest.
Money	Any generally accepted medium of exchange which enables a society to trade goods without the need for barter; any objects or tokens regarded as a store of value and used as a medium of exchange.

Table 17.4 Determining the noun Sense

	Bank noun Sense 1	Bank noun Sense 2	Bank noun Sense 3
Deposit	0	0	1
Money	0	0	0

known as word sense disambiguation, used to eliminate the uncertainty of meaning in the polysemic linguistic unit.

One of the methods for disambiguation is quite simple: Given a target word that has multiple senses, we compare the senses to the senses of words located in the vicinity of the target word.

To disambiguate the term “bank” in “deposited money in the bank”, first assume a preprocessing step that clears the text from stop words, and eliminates plurals and inclinations. Next, a window surrounding the target word is defined. This window can be shifted sideways so more or less words are taken from before or after the target word. Suppose the result is the set of three words: bank, deposit, and money.

The following step is to capture the senses of the target word and the words in the window. Table 17.2 shows three senses of the target word. Table 17.3 presents senses of words in the window.

Finally, the comparison step identifies all common words in each sense of the target word and each sense of each of the surrounding words. The result is the sense of the target word that shares more words with any sense of any other word. The number of common words between the senses is given in Table 17.4. Consequently, the sense of the word “bank” in “deposited money in the bank” is the noun Sense 3, “Money, dealer table, counter, shop”.

When irony is present or when ambiguity is used for comedy, this may not work. Consider the example, “War does not determine who is right—only who is left”.

17.4 The Case Base Container

While the vocabulary container identifies and selects indexing terms, the case base container includes the methods that focus on the cases and case bases. One of the distinctions is that indexing and case vocabulary tend to focus more on problems whereas cases include problems and solutions. When both problems and solutions are in textual format, textual methods need to be considered for both as well.

This section covers more methods that allow the structure in text to jump from left to right in Fig. 17.1. Rather than focusing on the expressions to be known and used by the case-based reasoner, the methods in this section provide structure to the case base, by identifying and associating cases in case bases. These methods include ways to convert text into non-text structured forms so the standard CBR can be used, that is, when cases are represented by standard attribute-value pairs.

17.4.1 Hypertext

One step from free text to structure is generating a hypertext structure. This would be a semiformal representation. Hypertexts organise text objects in a network in which they are connected by edges called hyperlinks. The associative representation form avoids the linear form of free texts. It also improves meaning because it adds more information to the associated concepts, providing a form of structural knowledge representation. Hypertexts have also the character of being more natural, as one may want to learn about a concept's associations before moving on with the text linearly. This naturalness has the potential to improve understanding and retrieval.

Converting text to hypertext, called post hoc authoring, requires division of the original text into meaningful units as well as meaningful interconnections of the units. This division is the result of text segmentation, discussed in Sects. 17.2.2 and 17.5.3.

Given two consecutive segments, one wants to know how much they have in common. The segmentation should, in particular, observe a topic transition in the narrative. A basic assumption is that this is usually accompanied by two facts:

- Introducing new words,
- Stopping using previous words.

That can be done by finding out how many previous words repeated in the second segment. Therefore, a divergence measure would be effective in detecting the topic boundaries, because it measures these two facts equally. This can be done by applying the Kullback–Leibler model or the symmetric Jensen–Shannon measure (see Sect. 17.5.3). Also used as similarity measures, here they can be used to identify blocks in text in order to create hyperlinks and thus add structure to text.

Consider the use of Web pages from wikis for a case base. Common hyperlinks shared by two pages can be considered as information entities in case retrieval nets (see Sect. 17.4.3), introduced in Chap. 14, Advanced Retrieval. The idea that a hyperlink is an indexing unit can facilitate dynamic creation of cases as well as similarity assessment.

17.4.2 Information Extraction

Information extraction (IE) originated as a methodology to extract content from texts to populate databases. The IE system must be input with domain guidelines that specify what to find and what to extract. In performing their task, IE systems analyse the text, searching for portions that might contain the sought information. In an IE task, a number of slots to be filled are specified, together with either a limited set of possibilities for descriptors for each slot or some specification for open-ended values. The values for the slots are strings from the source text. Natural language processing techniques are typically employed to allow word recognition and context understanding.

One variation from standard natural language processing techniques is the use of linguistic patterns in knowledge-based IE. These patterns can be inductively learned and then used to guide information extraction. Although inductive learning is automated, it may be necessary to manually select an initial set of training instances.

Another variation combines linguistic patterns with the existing structure in the texts. Here, we refer to structure as rhetorical patterns that can be highly predictive of text contents (see Sect. 17.2.2). One typical occurrence of texts that present this characteristic is when they are composed by professionals who have similar backgrounds. Sometimes, organisations will require texts to adhere to formatting requirements in order to be included in a database so as to increase their comprehension by the general public. When texts are this tamed, linguistic and other patterns can be identified without induction and then used for the purposes of information extraction. Moreover, when the language of the texts is well known, this template mining method can use lexicons to guide extraction.

Typical information extraction methods use two parameters to guide the extraction of a value to be assigned to an attribute. The first parameter is the text window, which embeds patterns that determine their localization, for example, a paragraph containing a certain combination of linguistic patterns or simply a heading.

The second parameter is the value to be extracted. Once the window is found, a list of potential expressions may be searched through simple matching, or natural language processing techniques may be needed for an explicit meaning to be obtained.

Using IE, virtually any text database can be converted into a case base. The main requirement is that we know that each record consists of a problem-solution context. Other requirements include knowing the domain for selecting the indexing vocabulary. Converting the text database into a case base becomes purely an IE task.

Consider, for example, those daily etiquette advice columns one finds in most newspapers. They are small texts that have clearly distinct problem and solution descriptions. Examples of features to be extracted would be the third person involved, his or her relationship with the advice seeker, and the subject of the issue. The solution might even be kept as a textual feature called *advice*. To extract the relationship between the advice seeker and third person involved, a simple search for words designating relationship can provide some reasonable initial results (e.g., friend, mother-in-law, cousin). As more accuracy is needed, more sophisticated IE can be used.

17.4.3 Information Entities in Basic Case Retrieval Nets

Information Entities (IEs) are indices that compose an indexing vocabulary for a CBR system. As you may remember, the indexing vocabulary is the set of terms that are expected to appear in all cases and that may have some predictive power with respect to the domain (see Chap. 6, Basic Similarity Topics, for more on this). The goal of the indexing vocabulary is to guide similarity and retrieval. This is how IEs can be used: First, IEs must be made general so they will not be limited to their surface form. For example, take a term from a source case, and identify its grammatical variations (e.g., plurals, verbal inflections, variations through prefixes and suffixes) and its abbreviations, synonyms, and translations; assume the word is *friend*, for instance include buddy, pal, acquaintance, and so on.

Second, they must be structured as nodes in a network of cases. These are the so-called Basic Case Retrieval Nets, presented in Chap. 14, Advanced Retrieval.

The steps for using IE and CRN are as follows:

- (1) Manually identify task-specific terms.
- (2) Enhance the similarity power of the IEs by contextualizing them through domain-specific ontologies and a generic thesaurus.
- (3) Connect IEs as nodes linked by their meaning with similarity arcs.
- (4) Create the retrieval algorithm to propagate activation through the network.

17.5 The Similarity Container

Intuitively, similarity between texts means that they have something in common. What the “something” is depends on the intended utility. However, a serious problem is that these commonalities may be hidden in the text and not be easy to extract. They can be related to the information content or to what is relevant.

This section examines similarity measures that can be used for texts and their representations. We start by describing relevance-oriented measures, structure-oriented measures, and measures for segments; then we specify the methods that compare queries to documents, and finally methods tailored for text-to-text comparison. As in the case base container, methods in the similarity container may address both problem and solution aspects.

17.5.1 Relevance-Oriented Measures

Weights denote relative relevance and are usually obtained by frequency or predictive ability. This has been described in Chap. 6, Basic Similarity. In documents, weights can be chosen as a variation of:

- (i) Term frequency, f_{ij} , the number of occurrences of term y_j in document x_i ; and

(ii) Inverse document frequency (IDF), the weight of the query term q_i . Often it is computed as:

$$\text{IDF}(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5},$$

$g_j = \log(N/d_j)$, where N is the total number of documents in the collection and $n(q_i)$ is the number of documents containing q_i . The result is that this measure induces, as expected, a similarity ordering of the documents; this ordering is often called a ranking.

An alternative approach to weighting documents is probabilistic. The relevance of a document d is denoted by R and the non-relevance by $\neg R$. The idea is to rank documents based on the conditional probability $\text{Prob}(R|d)$ that document d is relevant, given its presentation. This probability cannot be calculated directly. The application of Bayes' theorem results in the following formula:

$$\text{Prob}(R|d) = \frac{\text{Prob}(d|R)\text{Prob}(R)}{\text{Prob}(d)}.$$

Here the naïve assumption that the terms in the document occur statistically independently is made. This allows us to define the weight function ω for attributes a , $\omega(A) = a$, denoted by $\omega(A = a)$. For convenience, we assume $\omega(A = 0) = 0$.

When documents are represented with conditional probabilities, the similarity function between a query q and a document d can be defined as:

$$\begin{aligned} \text{sim}(A = a) &= \log \left(\frac{\text{Prob}(A = a|R)\text{Prob}(A = 0 | \neg R)}{\text{Prob}(A = a|R)\text{Prob}(A = 0 | R)} \right), \\ \text{sim}(q, d) &= \sum_a \text{sim}(A = a). \end{aligned}$$

17.5.2 Structure-Oriented Similarity Measures

Words can be considered similar by virtue of multiple aspects. The strongest level of similarity between words is when they are synonyms, and thus they are interchangeable enough to preserve the same meaning of a sentence. Semantic similarity is a looser concept that relates two entities by sharing ontological parent or child relations. Structure-oriented similarities are semantic similarities computed with a semantic structural representation such as a taxonomy or ontology (see Chap. 12, Advanced CBR Elements, for more on ontologies). In these structures, concepts can have multiple parents as well as multiple children along *is-a* relationships.

The similarity measures using structures can be distinguished into two categories:

- (1) Path-oriented measures: They have been discussed in Chap. 6, Basic Similarity Topics.

- (2) Content-oriented measures: They are based on measurements of the information content of each concept. This is defined as the number of times each concept, or any child concept, occurs in the corpus. It can be expressed as a probability in such a way that the information content of each node increases monotonically towards the root node, which will have an information content of 1.

The information content $IC(c)$ of a concept c is formally defined as:

$$IC(c) = -\log \text{Prob}(c),$$

where $\text{Prob}(c)$ is the probability of encountering an instance of concept c in the corpus. IC is standard in information theory.

This gives rise to information-oriented measures of similarity. Their difference with all the measures discussed in Chap. 6, Basic Similarity Topics, is that they do not consider syntactic aspects and weights assigned by the designer or user only, but also parts of the content, namely, the information content.

For two objects o_1 and o_2 , $P(o_1 o_2)$ is the set of parental concepts shared by both o_1 and o_2 . We take the minimum probability $\text{Prob}(o)$, denoted by p , if there is more than one shared parent. The minimum of these probabilities is

$$p_m(o_1, o_2) := \min(p(o) | o \in P(o_1 o_2)).$$

Hence, this is the measure with respect to DCP, the deepest common predecessor of the two, as introduced in Chap. 7, Complex Similarity Topics.

Some of the measures are introduced now. Their principles, benefits and limitations are then summarized in Table 17.5.

The Leacock–Chodorow Measure:

$$\text{sim}_{\text{LeaCh}}(c_1, c_2) = \log(\text{path_length}/2D),$$

where c_1 and c_2 are the concepts and D is the depth of the taxonomy.

It is based on simple edge counts in the *is-a* hierarchy of WordNet and deals with nouns only. The path length D is scaled by the depth of the taxonomy. This still does not consider information content; it is a special path-based measure, as discussed for taxonomies. Some related measures follow.

The Resnik measure:

$$\text{sim}_R(o_1, o_2) = IC(p_m(o_1, o_2)).$$

Hirst–St-Onge Measure:

$$\text{sim}_{HS}(c_1, c_2) = IC - \text{path length} - k \times d.$$

The Hirst–St-Onge measure requires the path length not be too long and not change direction too often.

The Lin measure:

$$\text{sim}_L(o_1, o_2) = 2IC(p_m(o_1, o_2)) / (IC(o_1) + IC(o_2)).$$

Table 17.5 Comparison of measures

Name	Principle	Pro's	Con's
Leacock–Chodorow	Finds the shortest path between concepts	Corrects for depth of hierarchy; Simplicity	WordNet nouns only; Is-a relations only
Hirst–St-Onge	Finds the shortest path between concepts	Considers changes in direction in path	WordNet nouns only
Resnik	Information Content (IC) of the DCP	Uses empirical information from corpora	Uses IC of concepts; only from DCP; WordNet nouns only; Is-a relations only
Jiang–Conrath	Extensions of Resnik scale DCP by IC of concepts	Accounts for the IC of concepts	WordNet nouns only; Is-a relations only
Lin	Extensions of Resnik by IC of concepts	Accounts for the IC of concepts	Same as Resnik

The Lin measure uses both the information content of the shared parents and that of the query terms and adds a normalization factor to the Resnik measure.

The Jiang–Conrath Measure:

$$\text{sim}_{JC}(o_1, o_2) = (\text{IC}(o_1) + \text{IC}(o_2) - 2 \cdot \text{IC}(p_m(o_1, o_2)))^{-1}.$$

The probabilities in these measures are frequencies determined by counting in large text documents. Table 17.5 summarizes properties of the measures.

The Leacock–Chodorow measure is path-based while the other three are in addition also content-based. The utility of these measures is not directly connected with an application. The success is measured rather in terms of understanding the texts; this is an intermediate step for the overall success. These measures can make use of so-called context vectors (see Sect. 17.3.4).

17.5.3 Measures for Segments

Given the word segments W_1 and W_2 , the Kullback–Leibler distance is computed by

$$d_{KB}(W_1, W_2) = \sum_{i=1}^n q_i \log \frac{q_i}{p_i},$$

where $p_i = \text{Prob}(t_i | W_1)$ and $q_i = \text{Prob}(t_i | W_2)$ and t_i are the terms considered.

The Kullback–Leibler distance measures how many new words occur in the segment part while the Jensen–Shannon distance measures also how many new words occur in the second part. In this way one can measure the distance of the probability distributions of the words of the second text segment from the first one.

A simple way to construct segments is as follows:

- (1) Select a distance measure d and a threshold θ for the distance.
- (2) Divide the text into n very small consecutive segments W_i .
- (3) Compute the following:

```

For  $i = 1$  to  $n - 1$  DO
  Compute  $d(W_i, W_{i+1})$ 
  IF  $d(W_i, W_{i+1}) < \delta$  THEN join  $W_i$ , and  $W_{i+1}$ 

```

These segments can then be used as cases for further processing.

Two other measures, introduced in Chap. 6, Basic Similarity Topics, can be used to compare letter and word strings or vectors, and Hamming and Levenshtein distances. These are context-independent similarities that can also to compare two text strings.

17.5.4 Text to Text Similarity Measures

A natural way to compare texts is to find out what they have in common. This is roughly called the overlap between the texts. This will now be made precise in order to define similarity measures. Such measures need the similarity of words based on the relatedness between the words for modelling the semantic similarity of texts as a function of the semantic similarity of the component words.

An example of such a measure is the Lesk measure that was extended to the gloss overlap measure of Banerjee and Pedersen. This measure computes the overlap score by extending the glosses of the concepts under consideration to include the glosses of related concepts in a hierarchy.

Suppose two texts t_1 and t_2 are given that have n m -phrases in common.

Definition 17.1

- (i) $\text{overlap}(t_1; t_2) = \sum_n m^2$
- (ii) $\text{sim}_{\text{BP}}(t_1, t_2) = \tanh\left(\frac{\text{overlap}(t_1, t_2)}{\text{length}(t_1) + \text{length}(t_2)}\right)$

This similarity function incorporates two normalizations. The first one is with respect to the length of the texts. The purpose for the second one, taking \tanh , is to reduce the influence of outliers.

17.5.4.1 Cosine

In Chap. 6, Basic Similarity Topics, the scalar (or dot) product was introduced. This product defines the multiplication of two vectors, which represents the cosine of the angle between them. The cosine of the angle of two vectors provides a measure of

the expected similarity between them. This measure compares the vectors in a space of n unique terms that comprise the space.

$$(\vec{x}, \vec{y}) = \vec{x} \cdot \vec{y} = \sum_{i=1}^n x_i y_i.$$

Normalization is incorporated based on the length of the vectors. Then, the similarity between two documents d_1 and d_2 with n terms is given by:

$$\text{sim}(d_1, d_2) = \frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{|\vec{V}(d_1)| |\vec{V}(d_2)|}.$$

17.5.4.2 Query to Text Similarity

The cosine is also used to compute similarity between queries and documents. When assessing the similarity between the query and a document, $\text{sim}_{\text{vs}}(q, x_i)$ is typically used to reflect the relevance of the document to the query. The document weights w_{ij} and the query weights v_j are

$$w_{kj} = f_{kj} \cdot \log(N/d_j)$$

and

$$v_{ki} = \log(N/d_j) \text{ if } y_j \text{ is a term in } q \text{ and } 0 \text{ otherwise.}$$

The cosine has been used for text represented with most of the representations discussed under the vocabulary container. It is usually used for comparing texts represented with the vector space model, and texts represented with reduced and distributed representations.

Another measure used to compute query-text similarity is the one used in the system Okapi BM25:

$$\text{sim}(q, d) = \sum_{i=1}^n \text{IDF}(q_i) \cdot \frac{f(q_i, d)(k+1)}{f(q_i, d) + k \cdot (1 - b + b \cdot \frac{|d|}{\text{avgdl}})}$$

$f(q_i, d)$ is the term frequency of q in the document d , $|d|$ is the length of the document d in words, and avgdl is the average document length in the text collection from which documents are drawn. k and b are free parameters, usually chosen in the absence of an advanced optimization, for instance, k [1.5, 2] and $b = 0.75$.

17.6 The Adaptation Container

Adaptation methods for text become relevant when cases are not converted into attribute-value features but are kept in text form during retrieval. The following CBR steps consequently are still performed in text. Evaluation of a proposed solution that is in textual format includes metrics for the quality of the text and for the quality of

the solution in its ability to solve the input query. Adaptation methods for text need to manipulate the text that is both in problems and solutions.

This is typically done through rules, which are not easily amenable to text. When rules are needed for this step, methods that add structure both from a vocabulary and a case base perspective can be useful for identifying rule antecedents. When adaptation is required, it may involve yet another natural language task, that of composition.

Natural language processing techniques can be used to evaluate the overall quality of written text. Methods such as those of grammar and spell checkers used in text editing software are the simplest and most superficial. The needs will vary based on the quality of the original cases and on whether the solution was combined or reused in its entirety.

Ideally, texts to be adapted should be annotated with their goals. Goals may be used as antecedents, for example.

An example of an adaptation rule for textual cases is the drafting of a document. Suppose the problem query is a new client who asks for a quote on service maintenance for IT resources. The company that provides IT support uses the goals of the process to retrieve the case that contains the letter draft and the cost of the service. The reuse step first analyses the differences between the retrieved problem and the new query. For the differences, adaptation rules are triggered, determining which excerpts to remove and which to modify from the retrieved letter.

Pattern matching of both rule antecedents and rule consequents can be potentially used to match sentence annotations. These annotations need to correspond to antecedents of adaptation rules, which would work like a standard adaptation method, as discussed in the Chap. 9, Adaptation.

17.7 What Textual CBR Method Should I Use?

This chapter presents mainly two ways of handling textual knowledge in CBR: (1) Use representations tailored for text and reason by manipulating them; and (2) Convert textual knowledge into attribute-value pairs and use the methods discussed in Parts I, II, and III.

The most important decision when faced with textual knowledge for CBR is about which of the two alternatives to pursue. To make this decision, some aspects need to be taken into account.

Consider the problem-solution distribution in the documents. There are two main ways in which textual knowledge is presented for textual CBR. First, source texts may contain problems and solutions within them. In this situation, it is important to determine whether their descriptions are mixed. This is closely related to the need for adaptation. This will influence the decision on manipulating text or converting text into a structured form. If the distinction is hard, then the latter is preferred because textual representations would require the initial partition of the problem and solution parts before representation. These are somehow related as adaptation

requires knowledge both about problem and solution parts. It is easier to deal with adaptation if both problem and solution parts are distinctly represented.

Second, texts can be used as problems only and a solution needs to be acquired or be available separately. These are typical of interpretive CBR, where the solution is an interpretation of the textual document. This is a situation where using a textual representation may be beneficial. Utilizing a representation such as LSI and using the cosine for similarity may produce accurate similarity scores for interpretive CBR.

Consider the magnitude of the document collection, in terms both of text quantity and length. This would determine if any manual knowledge engineering is feasible before determining whether it is even desirable. Large collections of texts can rely on automated methods for processing texts and on machine learning methods to build representations, feature extraction for indexing, and clustering. If some manual engineering is feasible, then methods that require comprehension of text structures can be used.

Extremely large document collections can benefit from the use of reduced and incremental representations like random indexing, if not for reasoning, at least for an initial assessment of case distribution, for example.

Consider the nature of the texts in terms of language, domain, provenance, occurrence of proper nouns, importance of negation and ordering. The combination of the dimensionality of the text collection and these other factors will narrow down the selection of methods for preprocessing, representation, and retrieval. Though ordering may be addressed with holistic representations, importance of negation comprehension suggests language processing with syntactic analysis.

17.8 Tools

Few general CBR tools have special features to deal with textual CBR. One is jColibri (<http://gaia.fdi.ucm.es/projects/jcolibri/#Download>); see Recio-García et al. (2005).

There are several algorithms and packages available for stemming, for instance, the Porter Stemming Algorithm (<http://tartarus.org/~martin/PorterStemmer/>).

For random indexing see <http://code.google.com/p/airhead-research/wiki/RandomIndexing>.

SQL Server 2008 has built-in support to perform free-text matching through a function. The FREETEXTTABLE function computes the ranking of text match using the Okapi BM25 (<http://msdn.microsoft.com/en-us/library/ms177652.aspx>). Okapi BM25 is a ranking function used by search engines in information retrieval (e.g., Robertson et al. 1995).

WordNet is freely and publicly available for download (<http://wordnet.princeton.edu/wordnet/download/>).

The LFTToolkit converts parse trees into logical forms (<http://www.isi.edu/~hobbs/LFTToolkit/index.html>).

GATE Teamware (<http://gate.ac.uk/teamware/>) is an annotation tool. KEA is a tool to extract keywords from documents (<http://www.nzdl.org/Kea/>).

17.9 Chapter Summary

Textual CBR is used when knowledge sources for CBR knowledge containers are in textual form. There are two main strategies for addressing textual CBR. One is to convert any text into a machine readable form and manipulate it using methods discussed in the previous chapters. The other is to keep the original textual representation and adapt CBR methods to manipulate text.

When aiming to use text for problem solving, text needs to be recognised to the extent that problems and solutions can be identified, compared, adapted, and reused. We define text levels, properties, and problems to understand text, which makes textual CBR particularly challenging.

The vocabulary container for textual CBR is concerned with definition, delimitation, and representation of case language. The representations used in textual CBR include parse trees, n-grams, bag of words, vector representations, distributed, and reduced representations. The methods used to define the vocabulary container may include extracting features, enhancing the vocabulary with external sources or simply disambiguating vocabulary.

The case base container manipulates case bases from unstructured, semi-structured and structured texts. Semi-structured forms include hypertexts and case retrieval nets.

The similarity container for textual CBR can use a variety of measures to manipulate text. Many are meant to be used with linguistic ontologies like WordNet. It also includes measures typical of information retrieval.

The adaptation container for textual CBR is used when entire excerpts are adapted, as in document drafting. Not all representations are suited if adaptation is needed for textual reuse.

17.10 Background Information

Textual CBR became a frequently used term to denote methods that deal with texts in the design of CBR systems after the 1998 AAAI Workshop (Lenz and Ashley 1998). An initial overview of the area can be found in those proceedings. A recent overview is given in Weber et al. (2005).

Branting and Lester (1996) describe the design task of document drafting, which reuses adapted cases. They demonstrate rhetorical structures of self-explaining documents, a linguistic-based approach to identify functions in excerpts of texts. Initial work in speech acts has been introduced by Searle (1975). Cohen et al. (2004) also rely on the notion of this analysis to identify speech acts for classifying emails

according to intent. The rhetorical structure of a document has also been used to extract indices for textual CBR via template mining (Weber et al. 1998).

Jurafsky and Martin (2000) provide an extended and detailed reading on natural language and n-grams and on linguistics aspects including speech acts. Further reading on text segmentation can be found in Beeferman et al. (1999).

Vector representations are discussed in information retrieval texts such as Baeza-Yates and Ribeiro-Neto (1999) and Manning et al. (2008).

Latent semantic indexing is introduced in Deerwester et al. (1990). Plate (2003) introduces distributed representations in his book *Holographic Reduced Representations*.

Random indexing is discussed in Sahlgren (2006) and Kanerva et al. (2000). It was used in CBR in Öztürk and Prasath (2010). In Öztürk et al. (2010), the authors explain the use of HHR for representing word ordering.

Feature extraction is a vast field. Some works in textual CBR include Wiratunga et al. (2004), which describes feature extraction with boosting to induce decision stumps. In Wiratunga et al. (2006), the authors describe experiments with an unsupervised method for feature selection.

Seminal work in word sense disambiguation is by Lesk (1986).

Information Extraction for textual CBR is reviewed in Brüninghaus and Ashley (2001). Case retrieval nets are presented in Lenz and Burkhard (1997).

Seminal work in inverse document frequency is given by Spärck-Jones (1972).

Structure-oriented measures have been defined in the respective works by Leacock and Chodorow (1998), Resnik (1995), Hirst and St-Onge (1998), Lin (1998), and Jiang and Conrath (1997).

Additional measures we discuss are further presented in Kullback and Leibler (1951), Jensen (1996), Hardy et al. (1952), Levenshtein (1966), Hamming (1950), Lesk (1986) and Banerjee and Pedersen (2002).

Textual adaptation in CBR has been done by Branting and Lester (1996) and Lamontagne and Lapalme (2004). Wilson and Bradshaw (2000) proposed the use of the cosine to assess similarity between textual case features.

The Okapi method is discussed in (Robertson et al. 1995).

17.11 Exercises

Exercise 1 Use WordNet to find similarities between words in chapter headings of two textbooks on the same topic.

Exercise 2 Disambiguate the terms in the following sentence: “She hung up the kitchen phone and reached for a date”.

Exercise 3 Identify a collection of texts that describe problems and solutions. Determine whether problem and solution descriptions are mixed together or clearly distinct. Describe the most appropriate method to implement textual CBR.

Exercise 4 Compute the cosine between vector representations of two papers from the reference list below.

Exercise 5 Which distance measures can be used to find the similarity between words in WordNet?

Exercise 6 Create a rule to adapt an email text written to a classmate to be sent to the dean of your program.

References

- Baeza-Yates R, Ribeiro-Neto B (1999) *Modern information retrieval*. Addison-Wesley, New York
- Banerjee S, Pedersen T (2002) An adapted Lesk algorithm for word sense disambiguation using WordNet. In: Gelbukh A (ed) *CICLing 2002: computational linguistics and intelligent text processing*. Third international conference, Mexico City, Mexico, 17–23 February 2002. *Lecture notes in computer science*, vol 2276. Springer, Berlin, p 136
- Beeferman D, Berger A, Lafferty JD (1999) Statistical models for text segmentation. *Mach Learn* 34:177–210
- Branting LK, Lester JC (1996) Justification structures for document reuse. In: Smith I, Faltings B (eds) *EWCBR-96: advances in case-based reasoning*. Third European workshop, Lausanne, Switzerland, November 1996. *Lecture notes in computer science (lecture notes in artificial intelligence)*, vol 1168. Springer, Berlin, p 76
- Brüninghaus S, Ashley KD (2001) The role of information extraction for textual CBR. In: Aha DW, Watson ID (eds) *ICCBR 2001: case-based reasoning research and development*. 4th international conference on case-based reasoning, Vancouver, BC, Canada, July/August 2001. *Lecture notes in computer science (lecture notes in artificial intelligence)*, vol 2080. Springer, Berlin, p 74
- Cohen WM, Carvalho VR, Mitchell TM (2004) Learning to classify email into speech acts. In: Lin D, Wu D (eds) *EMNLP 2004: empirical methods in natural language processing*, Barcelona, Spain, 25–26 July 2004. ACL, Stroudsburg, p 309
- Deerwester S, Dumais ST, Furnas GW et al (1990) Indexing by latent semantic analysis. *J Am Soc Inf Sci Technol* 41:391–407
- Hamming RW (1950) Error detecting and error correcting codes. *Bell Syst Tech J* 29:147–160
- Hardy GH, Littlewood JE, Pólya G (1952) *Inequalities*, 2nd edn. Cambridge University Press, Cambridge
- Hirst G, St-Onge D (1998) Lexical chains as representation of context for the detection and correction of malapropisms. In: Fellbaum C (ed) *WordNet: an electronic lexical database and some of its applications*. MIT Press, Cambridge, pp 305–332
- Jensen FV (1996) *An introduction to Bayesian networks*. UCL Press, London
- Jiang JJ, Conrath DW (1997) Semantic similarity based on corpus statistics and lexical taxonomy. In: *International conference research on computational linguistics (ROCLING X)*, Taiwan, 1997. <http://arxiv.org/pdf/cmp-1g/9709008.pdf>. Accessed 27 Feb 2013
- Jurafsky D, Martin JH (2000) *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition*, 2nd edn. Pearson, Upper Saddle River
- Kanerva P, Kristofersson J, Holst A (2000) Random indexing of text samples for latent semantic analysis. In: Gleitman LR, Joshi AK (eds) *Twenty-second annual conference of the cognitive science society*, Philadelphia, PA, 13–15 August 2000. Cognitive Science Society, Ann Arbor, p 103
- Kullback S, Leibler RA (1951) On information and sufficiency. *Ann Math Stat* 22(1):79–86

- Lamontagne L, Lapalme G (2004) Textual reuse for email response. In: Funk P, González-Calero PA (eds) ECCBR 2004: advances in case-based reasoning. 7th European conference, Madrid, Spain, August/September 2004. Lecture notes in computer science (lecture notes in artificial intelligence), vol 3155. Springer, Berlin, p 242
- Leacock C, Chodorow M (1998) Combining local context and WordNet similarity for word sense identification. In: Fellbaum C (ed) WordNet: an electronic lexical database and some of its applications. MIT Press, Cambridge, pp 265–283
- Lenz M, Ashley KD (eds) (1998) Textual case-based reasoning: papers from the AAAI-98 workshop. Technical report WS-98-12. AAAI Press, Menlo Park
- Lenz M, Burkhard H-D (1997) CBR for document retrieval—the FAILQ project. In: Leake DB, Plaza E (eds) ICCBR 1997: case-based reasoning research and development. Second international conference on case-based reasoning, Providence, RI, July 1997. Lecture notes in computer science (lecture notes in artificial intelligence), vol 1266. Springer, Berlin, p 84
- Lesk M (1986) Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In: SIGDOC 1986: 5th annual conference on systems documentation. ACM, New York, p 24
- Levenshtein VI (1966) Binary codes capable of correcting deletions, insertions and reversals. *Sov Phys Dokl* 10(8):707–710
- Lin D (1998) An information-theoretic definition of similarity. In: Shavlik JW (ed) ICML 1998: 15th international conference on machine learning, Madison, Wisconsin, August 1998. Morgan Kaufmann, San Mateo, p 296
- Manning CD, Raghavan P, Schütze H (2008) Introduction to information retrieval. Cambridge University Press, Cambridge
- Öztürk P, Prasath R (2010) Recognition of higher-order relations among features in textual cases using random indexing. In: Bichindaritz I, Montani S (eds) ICCBR 2010: case-based reasoning research and development. 18th international conference on case-based reasoning. Lecture notes in computer science (lecture notes in artificial intelligence), vol 6176. Springer, Berlin, p 272
- Öztürk P, Prasath R, Moen H (2010) Distributed representations to detect higher order term correlations in textual content. In: Rough sets and current trends in computing. Lecture notes in computer science, vol 6086. Springer, Berlin, p 740
- Plate T (2003) Holographic reduced representations. CSLI lecture notes, vol 150. CSLI Publications, Stanford
- Recio-García JA, Díaz-Agudo B, Marco A, Wiratunga N (2005) Extending jColibri for textual CBR. In: Muñoz-Avila H, Ricci F (eds) ICCBR 2005: case-based reasoning research and development. 6th international conference on case-based reasoning, Chicago, IL, USA, August 2005. Lecture notes in artificial intelligence, vol 3620. Springer, Berlin, p 421
- Resnik P (1995) Using information content to evaluate semantic similarity in a taxonomy. In: IJCAI 1995: 14th international joint conference on artificial intelligence, Montreal, Canada, 1995. Morgan Kaufmann, San Francisco, p 448
- Robertson SE, Walker S, Jones S et al (1995) Okapi at TREC-3. In: Harman DK (ed) TREC-3: overview of the third text retrieval conference. NIST special publication 500-226. NIST, Gaithersburg, p 109
- Sahlgren M (2006) The word-space model: using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces. Dissertation, Stockholm University
- Searle JR (1975) A taxonomy of illocutionary acts. In: Gunderson K (ed) Language, mind and knowledge. Minnesota studies in the philosophy of science, vol VII. University of Minnesota Press, Minneapolis, pp 344–369
- Spärck-Jones K (1972) A statistical interpretation of term specificity and its application in retrieval. *J Doc* 28:1–21
- Weber RO, Martins A, Barcia RM (1998) On legal texts and cases. In: Lenz M, Ashley KD (eds) Textual case-based reasoning: papers from the AAAI-98 workshop. Technical report WS-98-12. AAAI Press, Menlo Park, p 40

- Weber RO, Ashley KD, Brüninghaus S (2005) Textual case-based reasoning. *Knowl Eng Rev.* 20(3):255–260
- Wilson DC, Bradshaw S (2000) CBR textuality. *Expert Update* 3(1):28–37
- Wiratunga N, Koychev I, Massie S (2004) Feature selection and generalisation for retrieval of textual cases. In: Funk P, González-Calero PA (eds) *ECCBR 2004: advances in case-based reasoning. 7th European conference, Madrid, Spain, August/September 2004. Lecture notes in computer science (lecture notes in artificial intelligence)*, vol 3155. Springer, Berlin, p 806
- Wiratunga N, Massie S, Lothian R (2006) Unsupervised textual feature selection. In: Roth-Berghofer TR, Göker M, Güvenir HA (eds) *ECCBR 2006: advances in case-based reasoning. 8th international conference, Fethiye, Turkey, September 2006. Lecture notes in computer science (lecture notes in artificial intelligence)*, vol 4106. Springer, Berlin, p 340

Chapter 18

Images

18.1 About This Chapter

This advanced chapter addresses topics that are of particular concern to readers dealing with images occurring as queries or in the representation of cases. In domains like medicine or geography, images render large amounts of important data that are easily accessible. Because our interest is in CBR, this chapter focuses on problems concerning the understanding of images rather than discussing image processing techniques. In particular, we discuss image descriptions, semantics, similarity, retrieval, and cases. Understanding of image processing is not required but some knowledge of image understanding is helpful.

18.2 General Aspects

The saying *an image is worth more than a thousand words* puts forth the idea that images carry a lot of information that can be easily understood. In fact, the human eye can recognise a vast amount of information by looking at an image and interpret it as conveying a message as substantial as a thousand words. Consequently, analogously to the difficulties of capturing the meaning of a thousand words in text, here in this chapter we face the challenge of interpreting images. In order to be able to interpret and associate a meaning with images we will introduce a level structure that can represent different levels of abstraction.

There is a huge number of image processing methods. These methods are concerned with processing the image, given as a pixel set, which we discuss below. For example, one can detect that a triangle is the image. In certain situations this can be satisfactory for solving the problem. However, the user simply calls such a procedure without understanding it. This is the view on image processing we have in this chapter.

This motivates the use of images in CBR. We can use images in queries as well as in answers and for this we can call image processing methods.

One utilizes images in different disciplines for different purposes, typically requiring different methods of generation and interpretation. There is usually no clear boundary between these methods. As an example, in medicine one can use an image for diagnosis and therapy as well. One can use an image of a car for advertising and for technical purposes as well. One can ask what pathology is visible in a presented image and one can also show an image with a typical pathology; this concerns knowledge hidden in an image also used for reasoning. These aspects are explained in detail: How are images used in CBR? We will also discuss the special problems arising when images are used as a “representation language”. In connection with images one has to observe that there is a huge number of image processing methods. Their use in this context is as follows:

- One has to know for what purpose one wants to use them;
- One does not have to know how they work.

This is analogous to the treatment of programming languages, where one also does not have to know the internal details. These methods are stored in some database, as a recommended database. Then one can call them on demand. We discuss methods related to those in Chap. 23, Relations and Comparisons with Other Techniques.

An important point is that images occur in the CBR reasoning process. They allow us to use knowledge hidden in images as when dealing with text. This raises the question about what is special to CBR when images are considered.

Instead of giving a formal definition of this, we repeat the main points connected with the use of the CBR process model:

- The use of the knowledge containers.
- In the context of images, one finds:
 - Problem-solution pairs.
 - Extended problems that ask for direct solutions as described in Chap. 3, on extended view.

In our discussion we follow the patterns provided by the CBR cycle and the knowledge containers. We allow that images occur in both

- Queries
- Solutions

That means that we use the process cycle as well as the knowledge containers as before. What is changed is the description language for questions and answers. Images can also be used when the cases are products. This happens regularly for products used in everyday life. Therefore, there is a particular relation to the first step in the CBR process cycle. This step is the process formulation. If one presents an image in a query, it has to be made clear what one wants to know. In an attribute-value representation this is usually clear because the query is stated explicitly. When using images it is hidden and depends on the context. Definitions of similarity measures are problematic if they refer to utility. Utility requires understanding. Because the computer understanding of images is fundamentally different (as well as for tex-

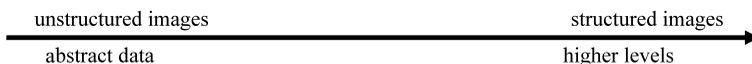


Fig. 18.1 From unstructured to structured

tual representations) we need a new approach (that is parallel to understanding of text, measured data, and speech).

All functionality of computational machines is programmed bottom-up, starting at the bit level. As a result, a computer has no means to generate or to interpret any meaningful images, unless someone implements the relevant methods.

There are different ways to associate attributes with images. For an example, later in this chapter we present Fig. 18.5, where we indicate different representation types besides attribute-based ones. In Chap. 5, Case Representations, Fig. 5.2, we pointed out that for each representation type one wants to transform it ultimately into an attribute-based representation. The introduction of levels is to support this.

In images one can represent an object in many ways. For instance, there is no unique way to represent a car. The heterogeneity of the representation forms makes the definition of the similarity measures quite difficult.

We present a level approach to understanding images that starts at the level of pixels and builds repeatedly to higher and increasingly more comprehensive levels. The presented approach is also an expression of the relationship of local versus global information in the context of images.

Humans can refer to higher levels of abstraction that are not readily available to computers. Moreover, while making decisions, humans can typically focus on these higher levels of understanding. They are not troubled by cognitive tasks at lower levels encompassing, for instance, basic signal processing—an eye sees without being told. The same applies to speech. In order to be able to interpret and associate a meaning with images, we will introduce a level structure that can represent different levels of abstraction. Levels are already known to us, for instance, from Chap. 17, Textual CBR.

The computer cannot immediately operate at the highest level of abstraction when confronted with an image. In order to empower the computer to interpret an image, i.e., to associate meaning with an image, our hierarchical structure can represent different levels of abstraction.

18.3 Image Structure

As in text, the images can be more or less structured, as shown in Fig. 18.1.

18.3.1 Image Levels

Next, we give a short overview of the levels with respect to the fact that there is no sharp boundary between them, as indicated in Fig. 18.2. In Sect. 18.5 we discuss the levels in detail.

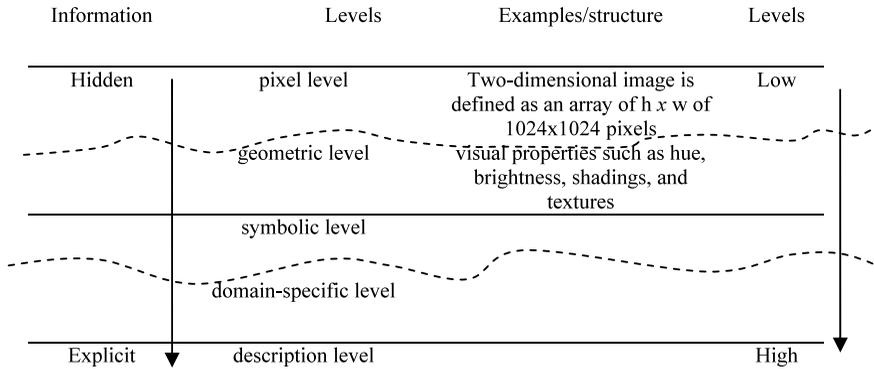


Fig. 18.2 Image levels

If the objects that appear in an image are known, that is, if precise mathematical descriptions are provided, standard image processing methods suffice for identifying them. If an image has to match by a simple similarity measure then this is a task for pattern recognition.

The situation is more involved if the objects are described imprecisely or even informally. Humans are still able to identify such poorly defined objects, as they consider typical and untypical features for a given overall situation. They are able to take complex contextual knowledge into account.

Queries and solutions, in turn, may revolve around the context of images.

(a) Typical examples where a query relates to an image Im are:

- (1) In medicine: Is it healthy or pathologic? What kind of disease does it show?
In traffic: What is the brand of this car?
- (2) In a landscape: What animal is it? Is there a pool?

In the first example type, one can use cases as (problem, solution) pairs. For the last two queries the extended CBR view usually works. The query is an X-ray image or an image of a car or an animal and the answer is a symbol denoting the brand or the disease type without referring necessarily to an experience. In general, the answers to such questions can be textual or in structural form, or contain images as well.

(b) An image Im may be part of an answer, further contributing to its solution. Such answers could be:

- (1) An image showing a specific cancer type.
- (2) A typical shape of a Volkswagen Golf.
- (3) A picture showing an aerial view of desert dunes.

As we can see, the forms of representation of queries and answers can vary quite strongly even when relying on non-compositional images. In addition to considering image composition, one could mark certain areas of interest in an image in order

introduce meta-information that guides queries and answers alike. This occurs frequently in medicine. The semantics of image data are multifaceted and speak to the viewer on various levels. This makes the definition of a similarity measure for finding the best (image-based) answer a fairly difficult task.

The most important challenges around image computation in our context are:

- Image processing.
- Image comparing.
- Image understanding.

In this chapter, we do not provide an introduction to image processing methods. Instead, we assume that we have a library of image processing methods that can be called on demand. We further assume that the reader roughly knows what kinds of methods exist, how they work, and what they accomplish. In some sense this presupposed library is a case base in its own right.

The situation is a little different in image understanding. This refers to semantics, and is therefore a major objective of this chapter.

18.4 The Level Structure

In order to build the understanding of images on a rigorous basis, we introduce a level-based data structure. The operators utilize image processing as well as symbolic computation methods and will be explained below. The levels of the introduced structure are heterogeneous, representing images in different ways at different levels.

All levels contribute to understanding the image data. Higher levels consider a wider context and host less specific but rather general, i.e., abstract, data. This tendency corresponds well with the idea of a global view, as opposed to a very specific, detailed local view. The following overview describes the hierarchy of levels realised by the level structure:

- (a) Lowest level: The **pixel level**. On this level the computer retrieves and maintains a digital image as an array of pixels. This level is analogous to the alphanumeric level in textual representations. At this level, any knowledge contributing to the understanding of the image data is still hidden in primitive, numeric data types. Ascending the level structure, the hidden information is made increasingly explicit, ultimately being represented in a symbolic language. That means, from level to level the obtained information from the image is more and more useful.

For standard image data, we distinguish the two-dimensional and the three-dimensional case:

- The syntax of the two-dimensional image is defined as an array of $h \times w$ (e.g., $1,024 \times 1,024$) pixels where a pixel is a grey value, a colour value or an element of $\{0, 1\}$ (black, white). A pixel is denoted as a_{ij} .

- The syntax of a three-dimensional image is defined as an array of $h \times w \times d$ (e.g., $1,024 \times 1,024 \times 64$) voxels where a voxel is a grey value, a colour value or an element of $\{0, 1\}$ (black, white). A voxel is denoted as a_{ijk} .
- (b) The **geometric level**: Here, elementary geometric objects are introduced as lines, curves, areas with their boundaries and brightness, and segments. These objects have visual properties such as hue, brightness, shading, and texture. Many of these properties are represented as attribute-value pairs. It has to be taken into account that mathematical objects are mere approximations of real objects. In addition, they do not occur as exactly mathematical objects and they do not satisfy precisely the mathematical definition. Some real-world occurrences are vaguely defined; for instance, “ovals” are informal variants of ellipses.
 - (c) The **symbolic level**: At this level, symbols are introduced for visually recognisable forms. These forms either are constructed at the pixel level or they are compositions of geometric objects.
 - (d) The **domain-specific level**: This is the lowest level of the domain. The objects are defined either symbolically or geometrically. For an example, see Fig. 18.4.
 - (e) The overall **description level**: At this level the meaning of the image is formulated in such a way that it provides an answer to the original problem. For instance, this image shows an unusual appearance of a part of the lung.

As mentioned, there is no clear line between the symbolic and the domain-specific levels.

Each level relies on specific representation methods and has specific similarity measures. The representation concepts of higher levels are obtained from those of lower levels by computational methods.

The levels are connected by certain methods, mostly from lower to higher levels. However, when going to a higher level one occasionally needs access to lower levels. We call methods steps. Before we go to details, we give a general recipe for creating the next higher level as follows.

- (1) Apply mathematical functions at the given level, for instance, image processing methods or filtering functions.
- (2) Apply feature (attribute) extraction methods that can be applied for obtaining a finite set of geometric objects.
- (3) Translate the obtained features into a symbolic description.
- (4) Combine the descriptions in order to obtain a description that is meaningful in the context of the application domain.

Steps (1) and (2) abstract the provided information, whereas steps (3) and (4) integrate the abstracted information and make it accessible and upward-compatible in the level hierarchy. Abstraction and integration in the steps (3) and (4) are crucial for rendering hard real-world problems computable and for providing the means for categorization and of comparison. For instance, one would not compare two ellipses by comparing their infinitely many points but rather by a similarity measure that operates on their parametric features. All four steps will now be illustrated by examples and discussed in more detail.

As a result, the user of a CBR system has to be able to formulate the requirements of the methods, addressing purposes of the user (i.e., what?) as well as the concrete implementations (i.e., how?).

18.5 The Image Pixel Level

Humans have no particular relations to sets of pixels as such. However, in the digital world, sets of pixels represent images. We observe that there are different kinds of pixels:

- (a) Single bits, 0 or 1 (e.g., black and white).
- (b) One-dimensional real number scalars from (0, 1) (e.g., grey values).
- (c) Multi-dimensional vectors to represent colour values (e.g., HSV or RGB).

The images are not an exact representation of reality, because of noise and necessarily inexact means of representation. However, there are several methods for simplifying or denoising images. These methods are a suitable example of simplification as an important means for reaching higher levels of abstraction.

There are two major kinds of simplification:

- (a) Segmentation.
- (b) Application of simplifying operators.

Modern segmentation methods offer numerous control parameters. It is important to utilize them in order to obtain several attributes associated with the given level, which will be used for further processing.

Simplification operators are discussed in the medical context in this section. There are numerous image processing methods that may be applied at the given level. Their purpose is the generation of objects that are useful at higher levels. Some of these methods are summarized in Table 18.1.

Even objects can belong to different levels, as the bounding box makes clear. Texture-related image processing methods are more intuitive. There are some well-established methods that provide valuable information. They refer to the arrangement and frequency of tonal variations, in particular, areas of an image. Most interpretations of the term texture have in common that they denote a grey value function on small areas that are often repeated. Textures are thus mainly used to quantify properties of the grey-level differences (contrast).

Several additional methods are utilized to perform extensive analyses on the pixel level. Textures are related to histograms. The statistics are concerned with the distribution of frequencies of grey (or colour) values. The *histogram* contains the probabilities $\text{Prob}(i)$ with which a grey value i occurs in the image in relation to the total number of pixels. Histograms can easily be visualised and be compared to one another by means of distance measures, most of which are presented in Chap. 6, Basic Similarity Topics.

A special topic is dimension reduction. Suppose certain patterns are given as functions $S \rightarrow U$.

Table 18.1 Some image processing concepts

Name	Meaning	Computed
Gravity centre	Centre of mass of a segment; used by distance computations: if the centre lies outside of the bounding box of a label, this label cannot be (semantically) associated with the segment.	xy-point
Segmentation	Dividing the image into blocks or clusters of pixels based on the pixels' properties. Deleting or adding details.	pixel clusters
Simplifying operators	Rectangle parallel to the $x - y$ axes, describing the convex hull for an object.	pixel data
Bounding box	Rectangle of the smallest size including the geometric object.	geometry
Grey value	Mean grey value of an image or an image segment. Can be either in HU values or in absolute pixel values.	real value
Texture	Grey level differences as contrast, area where change occurs, directionality or lack of it.	parametric features

For any classifier $C : U^S \rightarrow \{1, \dots, n\}$ and any subset $X \subseteq S$ we define an equivalence relation $=_{C,X}$ over U^S by putting for $\alpha_X, \beta_X \in U^X$:

$$\alpha_X =_{C,X} \beta_X \Leftrightarrow \text{for all } \gamma_Y \in U^Y : C(\alpha_X \cup \gamma_Y) = C(\beta_X \cup \gamma_Y), \quad \text{where } Y = S/X.$$

The influence potential (see Chap. 2, Basic CBR Elements, on influence relation) is then defined by

$$v_C(X) := |U^S / =_{C,X}|.$$

The purpose of the influence potential is to see how far a subset of values of the input pattern determines the class of the pattern. Subsets with low influence potential can be omitted. This reduction simplifies the computation of the similarity measure. In CBR, it simplifies also the similarity assessment.

Queries to this base result from the demands aforementioned. The similarity between the queries and the methods incorporates knowledge the methods can do as intended. This realises the view that one needs to know what one wants to achieve but not how this is done.

18.5.1 The Geometric Level

Here, elementary geometric objects are introduced as lines, curves, and segments. These objects have visual properties such as hue, brightness, shading, and texture.

Humans are able to recognise geometric objects unless they are unfamiliar or too complicated. Therefore, several conditions should to be met to facilitate object recognition. In particular, the objects should have the following properties:

- (a) The geometric objects should be recognisable as triangles, circles, rectangles, and other common shapes.

Fig. 18.3 Polygons for an oval 2D image

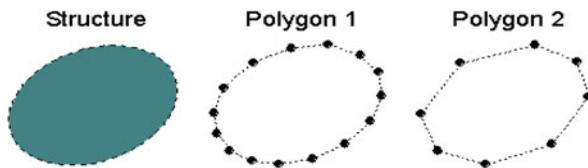
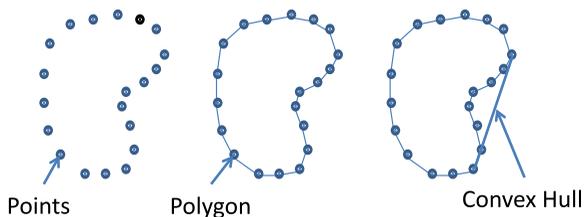


Fig. 18.4 Deformation of an oval



- (b) The objects should be structured in a clear way, composed of a few elementary structures.
- (c) The composite objects should not be too complex.
- (d) Additional properties like colours or textures should be made visible.
- (e) The objects should have an attribute representation that corresponds to intuitive descriptions and that can be recalled easily.

The definitions of the geometric objects as well as their properties and their associated methods are collected in another base: The geometric object base. It contains the methods to generate the geometric objects one uses.

A typical query to the geometry case base is:

How do we compute the intersecting area of two ellipsoids?

One has to acknowledge that in application domains such as medicine, objects at the corresponding level occur not as mathematically defined objects (like ellipses and rectangles) but as approximations of real-world objects. For instance, an object could be “almost a square”. Also, some of them are vaguely defined, such as “ovals”, which are informal versions of ellipses, as we see in Figs. 18.3 and 18.4. Nevertheless, the geometric object level may serve as a general orientation for application domains beyond imaging, to identify, abstract, and integrate objects that are concisely or fuzzily defined in mathematical terms. The question now is, how are the geometric objects represented to meet the above requirements? There are three kinds of criteria:

- They need to reflect their degree of uncertainty,
- Provide easily accessible data, and
- Be visualisable.

Varying degrees of fuzziness poses a challenge. At the end of the decision-making process, the degree of fuzziness can be very informative and lead to a decision. But when processed at lower levels, it leads to considerable information loss and is hard to visualise directly.

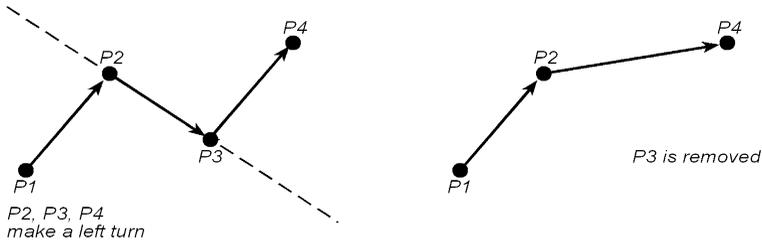


Fig. 18.5 Removed edges

Here, we propose approximate descriptions in terms of polygons and their properties. Polygon approximations are widely investigated in mathematics and image processing. There are four reasons for using polygon approximations:

- (a) Polygons are finite objects that can be easily stored in a computer.
- (b) With polygons one can approximate familiar geometric objects with a high degree of accuracy.
- (c) Polygons can be visualised.
- (d) With polygons we can associate properties that are considered by humans when discussing images.

In Fig. 18.3 we illustrate the use of polygons for approximating an oval structure. Polygon 2 sufficiently approximates the oval structure. It requires fewer computational resources than Polygon 1, as it requires merely eight vertices instead of 15.

Each of the two polygons is an approximation of the oval structure on the left-hand side.

In the next example (Fig. 18.4) we encounter some difficulties because of deformations of standard objects. Such deformations are quite important for a human viewer. In a verbal description a human would refer to the deformations and mention where they are (for instance, left or right) and how many one can see. Again, a polynomial approximation is used.

The convex hull of the polygon does not adequately reflect the deformation of the oval. If the form of the deviation conveys important information, it will be missed. Therefore, one should have methods for detecting and describing deviations of an ideal oval. This can be done by recording and counting turns or changes in the direction between pairs of vertices as Fig. 18.5 shows.

The left and the right polygons are quite different. Therefore, one cannot simplify the left figure by leaving out the point P3.

The turns are counted and expressed in terms of attributes of the shape. The length of the edges could, for instance, be utilized to denote another property. In this way, a distorted oval can be characterised through its deviating properties from an ideal oval.

Polygon approximations of objects and their properties can be understood as descriptive features of an image. The technique of describing deviations from ideal mathematical objects can also be used for other geometric objects. Examples are

ornaments, half-moons or crosses. They are basic elements of a so-called visual language. Such symbols are used to establish communication.

In summary, a geometric object has a twofold description:

- (1) The description of an ideal mathematical object represented as a formula.
- (2) The description of its deviation from the ideal shape.

Humans associate geometric objects with abstract meaning, which leads us to the next level.

18.5.2 The Symbolic and Domain-Specific Level

This level hosts domain-dependent descriptions. One kind of description is concerned with the special components of a given image, the other kind addresses the relationships among objects.

In images, spatial relations deserve a lot of attention. As a two-dimensional image can contain information about a three-dimensional situation, the interpretation depends on spatial knowledge. This knowledge may be partially of fuzzy nature; see Chap. 15, Uncertainty. Examples of spatial relationships in images are:

- Absolute positions of objects and their parts, for instance, “in the upper left corner there is an object X ”.
- Relative positions: Object X is left of object Y .
- Size of objects (again absolute and relative).
- Distances between objects.

Definition 18.1 The spatial position description PD of an object describes its position in the image.

Especially in technical domains, there are more refined methods for describing spatial relations. In medicine, for instance, the Frankfort-plane describes spatial relations in the brain.

Individual components carry various kinds of knowledge that is often equipped with other information: Knowledge about texture, colour, shape, and so on. This knowledge is available on the geometric level. The definitions of these objects and their relevant properties are contained in the symbolic and domain-specific base. There are two description steps for characterising the aspects of interest from an application-oriented view:

- (1) Description of the individual objects.
- (2) Description of their relations to each other.

In medicine, the main task of image analysis is distinguishing normal objects from pathological ones.

First we present in Table 18.2 an example of a structure occurring in medical images medical experts look at. It is essentially described by the values of certain attributes, including the degrees of uncertainty. In medicine this structure is called “skull”.

Table 18.2 Structure of a skull

Skull:

Shape = oval, GreatConcav = 1.4, Tolerance = 12 %;

Density = hyperdensity, GreyLevelMean = 253, Tolerance = 10 %;

RelativeDensity = hyperdensity, Structure = Cerebrum, GreyLevelMean = 109;

Texture = Homogeneous, StandardDeviation = 6.1, Tolerance = 10 %;

Position = Centre, Centroid = 260–276, Tolerance = 05 %;

RelativePosition = Around; Structure = Cerebrum, Centroid = 259–277.

We will not explain the individual terms in this structure; in medicine they are standard. The description refers to a non-pathological skull. Its fuzzy character is expressed in terms of tolerances. If they are overstepped then it may be pathological.

A query to the base could be:

What is the definition of a skull?

The answer could consist of naming the attributes and their allowed values. The descriptions refer directly to the visual impression of the human. But one has in addition to take into account that one and the same object can have different representations depending on the medium used (X-ray, NMR, and so on). It should also be remarked that there is no clear dividing line with the geometric level.

Such objects have a direct meaning in medicine. They have names and are used by experts for describing a situation of interest. This will lead us to the highest level of representing an image.

18.5.2.1 Graph Representations

Graphs have a twofold character. On the one hand they can be formally described as in Chap. 5, Case Representations. On the other hand they can be visualised and give a visual impression to a human viewer. This double role can be used as an intermediate representation between a pixel image and an attribute-based description on the domain level. One can take advantage of this by defining translations in both directions.

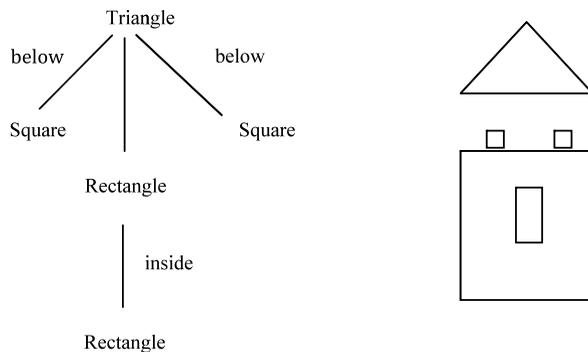
The starting point is to regard the use of polygons in arbitrary combinations of straight lines to represent real-world objects. As an example we consider a simple house. Its graph representation is in Fig. 18.6.

The left graph represents the geometric objects while the right graph serves for visualisation. These graphs can be annotated in different ways:

- coordinates, length, size,
- meaning (door, window, roof),
- colour,
- cost and so on.

The graph is of particular interest when one considers a special class of objects and their images. Then one can derive the properties of interest using the annotations.

Fig. 18.6 Graph representation of a house



18.5.2.2 Skeletons

The same technique can be used if humans sketch an image that they only vaguely recall. On the other hand, often they are able to draw a somewhat related graph. This technique is known as sketch-based modelling. The line structures are called skeletons or strokes. First, we show an example in Fig. 18.7.

On the left side an image of an animal is shown, on the right side a skeleton. The skeleton is a figure on the geometric level. It can either be mathematically defined as a graph or roughly drawn by a human. In the latter situation the mathematical structure has to be identified by image processing methods. See also Chap. 4, Application Examples.

The graph object can be handled and manipulated in a formal way, but one can also ask for the similarity between the image and the graph, for instance, the hand-drawn skeleton. In Sect. 18.11, Applications, a task for this area is shown. The case base can contain images of interest together with additional information or annotated graphs, depending on the application type.

18.5.3 The Overall Level

This level is the one that is ultimately of interest. On this level the meaning, i.e., the semantic of the image, is defined. Here the major objects of interest are located. In medicine these are, for instance, descriptions or representations of pathologies or non-pathologies. They are again collected in a base: The overall base.

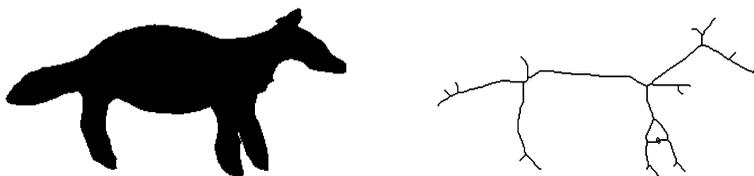


Fig. 18.7 Images and skeletons

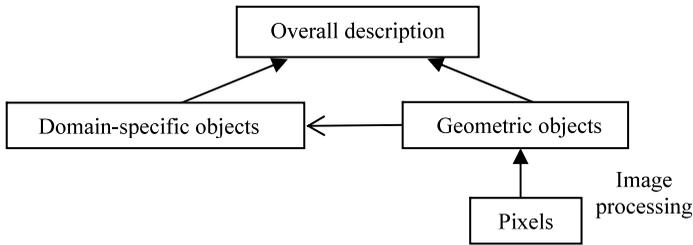


Fig. 18.8 Relations between levels

However, the base can contain (commented) images or symbolic descriptions and it may be split into subbases accordingly. The answer to the queries does not only depend on the domain but also on the pragmatics, i.e., the problem type. E.g., is the problem finding a diagnosis or a therapy?

The cases in a base can refer to other bases on lower levels. The overall level has access to the domain-specific and the geometric levels and these may in turn access the lowest level. On the overall level one detects geometric objects and symbolic and domain-specific objects. The tasks on this level are:

- (a) Identify objects of interest: What objects can one see on the image?
- (b) Locate them spatially in the image.
- (c) Determine attributes as size, texture, and so on. This refers to lower levels.

For these tasks one needs image processing methods again. If an image is called by a query, then the query will refer to these properties. Technically it means the answers to queries require access to all these levels. For instance, some levels can contain methods that operate on other levels and some may contain definitions from other levels. We see a very general influence diagram in Fig. 18.8.

The interplay between the highest level and the lower levels can be illustrated by a virtual phone conversation between a layman who sees the image and a medical expert who does not see it but has to make a diagnosis:

“Do you see a vertical symmetry axis and to the right and to the left of it two kidney-type symmetric darker forms”?—“Yes”;—“Is the distance from the axis for both forms about the same?”—“Yes”;—“Do both have the same size”?—“No, the left one is a little smaller”;—“How much smaller”?—“7.5 %”;—“Do both have the same brightness”?—“No, the left one is brighter”;—“How much?”—“12.5 %”;—“OK, now I know what the picture tells us”.

The human uses the visual competence of the eyes for giving the answers. If the answers are given by a software agent then this agent has to consult case bases that allow employing image processing methods stored in the base. The advantage of the software agent is that its answers are more precise than the one from a human who has to rely on visual impressions.

The example shows that understanding the meaning of an image is a process. The ordering of the questions in this example process reflects the way in which an expert

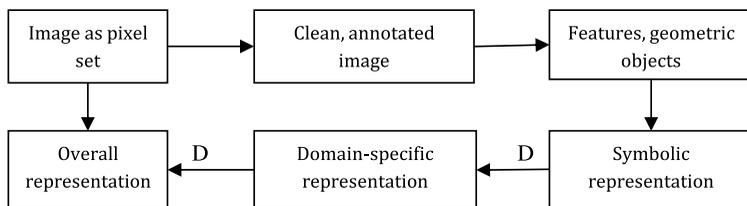


Fig. 18.9 The whole process

looks at the image. Due to impreciseness and the richness of the world, images do not always lead us to assigning them a precise meaning to the image. The rough set method suggests considering three possible objects of an image:

- (a) This is certainly a specific object of the real world.
- (b) This is definitely not a specific object.
- (c) This is in the uncertainty area, where we do not know exactly what the object is.

This can be formulated by using thresholds for the similarity measures as discussed in Chap. 15, Uncertainty; Fig. 18.9 summarizes the image understanding process.

Here the letter D denotes the application of domain and application-oriented knowledge; the vertical arrows denote image processing. The feature extraction has no prominent counterpart in the CBR cycle. The reason is that in attribute base representations the attributes are already given. They originated from problem solving before CBR came into the discussion and often even before computers were used. Compare this to the corresponding remarks in Chap. 19, Sensor Data and Speech.

So far the term meaning has been used in an informal way, the way humans use the term. However, for computer applications we need precise and formal definitions. This is the next topic.

18.6 Semantics

Image interpretation has not only to deal with the images as data structures but has also to take into account the domain of objects occurring on the images. Therefore, it makes reference to the outside world. This world is represented as a context.

To recognise an image means to understand what it means understanding is a process that can be modelled in steps. The main steps are:

- (a) Recognising domain-specific objects.
- (b) Relating these objects to each other.
- (c) Relating the image to the context.

Relating objects can be done in different ways, where geometric relations again play a role. For instance, one can

- identify spatial relations,
- compare shape, size, brightness, texture, and so on.

A real-world scenario S is represented in the image by a mapping:

$$Im : S \rightarrow \text{Images.}$$

If in the real-world scenario there are chairs and tables, these are mapped to certain image objects that represent them. The human associates a meaning with the image when looking at it and we want to formalize this process, at least partially. One can consider images as elements of a language because it is a means of communication. The elements are not letters, words or sentences, but rather pixels and geometric and domain-specific objects.

Following the local-global principle we proceed in the same way as for other complex objects. In this principle view we need to define and compute a function SF:

$$\text{SF} : \text{image} \rightarrow \text{meaning.}$$

The meaning is, following the usual procedure in logic, an interpretation in a model. This model is constituted by the mapping Im introduced below.

SF is called a semantic function. Going from local to global elements reflects the view of understanding as a process.

As always, we assume atomic components $(A_i, i \in I)$ and a constructor function C such that the whole scenario S is given by $S = C(A_i, i \in I)$.

An essential part of the constructor function C uses the part-of relation; the atomic components are given as values of attributes.

The semantic function SF has to reconstruct a description of S from the image, $Im(S)$, by analysing the representation of the image.

It is assumed that $Im(S)$ is also composed from atomic parts,

$$Im(S) = C_{im}(Im(A_i), i \in I).$$

Some essential parts of the constructor function are concerned with the geometric relations of the atomic parts. The formal definition of the semantics follows the identification of the real-world objects and proceeds in the following steps:

- (i) Identification of the atomic geometric objects.
- (ii) Assigning to certain geometric objects G some object $C = \text{SF}(G)$ such that $G = Im(C)$. This is the local semantics because it assigns a meaning to a part of the image. It compares to the semantics of atomic formulas in logic.
- (iii) If several of such objects occur in the image, the semantic function has to analyse their relative geometric positions and topological relationships in order to produce the position description PD. This corresponds to the logical connectives.
- (iv) From the steps above, the semantic function constructs the meaning as a global semantics of the image. This corresponds to the computation of the truth value in logic.

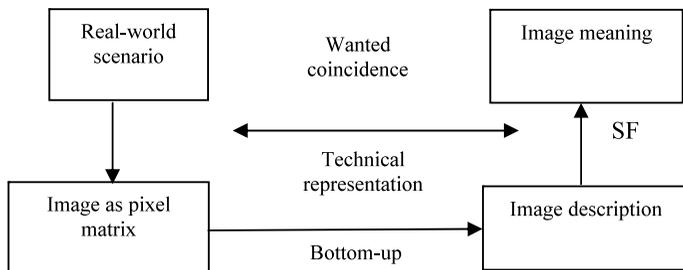


Fig. 18.10 Semantic interpretation

Fig. 18.11 CT image of the human head showing the fourth ventricle in the red circle



Hence the semantic function SF is decomposed in the following way:

$$SF(image) = SF(\{Im(C_i) | i \in I\}, PD),$$

where the C_i are the identified objects and PD describes their relations and positions. Figure 18.10 summarizes the whole process from the real world to the image semantics.

As a medical example we will introduce a simple image-understanding task. Firstly, the whole image of a human brain is in Fig. 18.11. It looks quite complex. A special part of image is marked as a region of interest that shows the fourth ventricle. We want to discuss how it can be identified.

The general task will be to understand the image of the brain and its different parts. Therefore, we have to identify parts of the image which correspond to anatomical structures of the human brain, in order to accept or reject some hypothesis about pathologies.

The image is represented on the pixel level. General medical knowledge tells us where objects of medical interest are located. One such location is marked with a red boundary in the figure: There is a horseshoe-shaped darker structure, which is clearly a candidate for representing some anatomical structure in the brain. As human experts we expect at this place the fourth ventricle. In fact, medical knowledge supports this. In an automated image-understanding task, this conclusion has to be achieved through a number of reasoning steps.

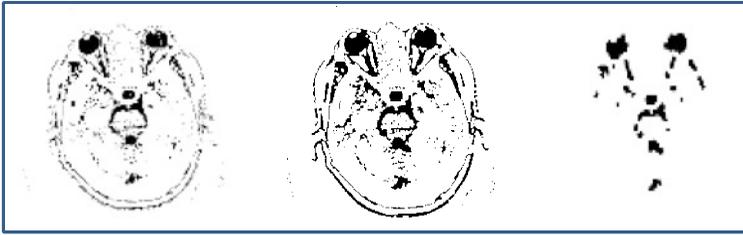


Fig. 18.12 Simplifying operators

On the pixel level the analysis starts with providing a basis for locating objects and parts that correspond to anatomical structures of the human brain. To locate parts, a method is to determine a coordinate system for the CT images of the particular patient. A specific coordinate system is the so-called Frankfort-plane. It is based on the positions of the centres of the eyes and the basis of the fourth ventricle. For this, image processing methods are available.

Next, the image has to be simplified in such a way that objects can be identified. One distinguishes between three main operators for simplification:

- The threshold operator omits all pixels with values above (or below) a certain threshold value.
- The closing operator preserves the rough parts exactly. Small missing parts are added.
- The opening operator preserves the rough parts in an approximate fashion; small details are omitted.

A medical image of the brain (the fourth ventricle) is shown in Fig. 18.12.

This example shows that a step for better understanding can still operate on a low level.

18.6.1 Aesthetics

Aesthetics influences human decisions for a great deal. For instance, the aesthetics of an object may easily overshadow price. The consequence for a solution provider is: What is aesthetics and how can we establish it?

Aesthetic properties are a special kind of semantics that apply in particular to images. Consider the situation where you are looking out of the window of your house in the evening and you see a car on the street. It is too dark to see details but you are quite sure that the car is a BMW. How can you do that? If you send an image to a computer can the computer do it too?

This is not easy to analyse. What one sees are larger parts with particular shapes. Usually one subsumes this under the concept of aesthetics; in another terminology this is described as typicality.

There are two views on this. One is described above: How can I identify an object just by looking at the typical features? The other view is: How can I design an object with typical features so that one can put it into a certain category (such as “my products”)?

For both purposes one needs to identify features on objects from the geometric level, which defines typicality or an aesthetic class. These features have to be extracted from objects that are usually defined by continuous real curves or planes. That means technical devices do not have technical properties only but also aesthetic ones and often the two are in conflict. As a result, the design process is quite complex. The main problems are:

- Designing an object which has a certain aesthetic property and satisfies other technological constraints. The latter are mandatory.
- Retrieving an object with a certain aesthetic property.
- Retrieving an aesthetic description of a given object.
- Designing a modification method for changing an object with respect to technological or aesthetic properties.

A first step is to define characteristics of objects from a human point of view, as for instance for cars:

- Sportsmanship.
- Elegance.
- Aggressiveness.

These are fuzzy concepts, but in the case base one may store reference objects (prototypes) that have these properties with membership degree 1 when we have a similarity measure sim such that, for a given object G and a prototype P for the property, the degree of G having the property P is $\text{sim}(G, P)$.

Next, we mention two crucial problems connected with aesthetics.

(a) First problem: Retrieval.

Example We have queries like “Give me all designs of ski-boxes developed by company XYZ which have a very sportive character without having an aggressive touch”. Therefore, the case base has to contain designs of the past, for example, in the form of CAD files or images, with an additional description of their design character. This is illustrated in Fig. 18.13.

(b) Second problem: Determination of the design character of a given object.

Example The query is: *What is the aesthetic character of ski-box A of company XYZ?* This is illustrated in Fig. 18.14.

The special terms determining a character are application-dependent and cannot be generated automatically. For instance, aesthetic properties are different for cars and wine bottles.

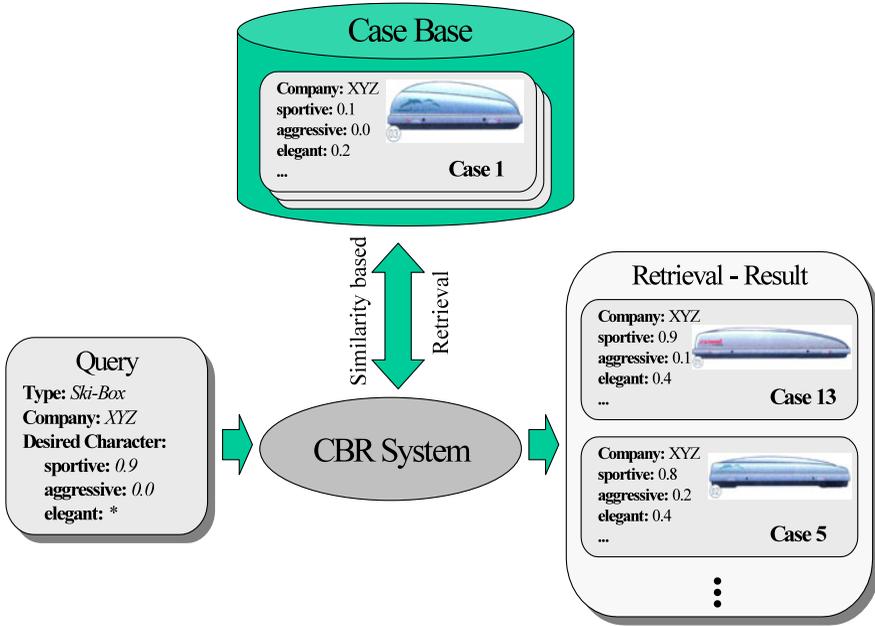


Fig. 18.13 Additional description elements

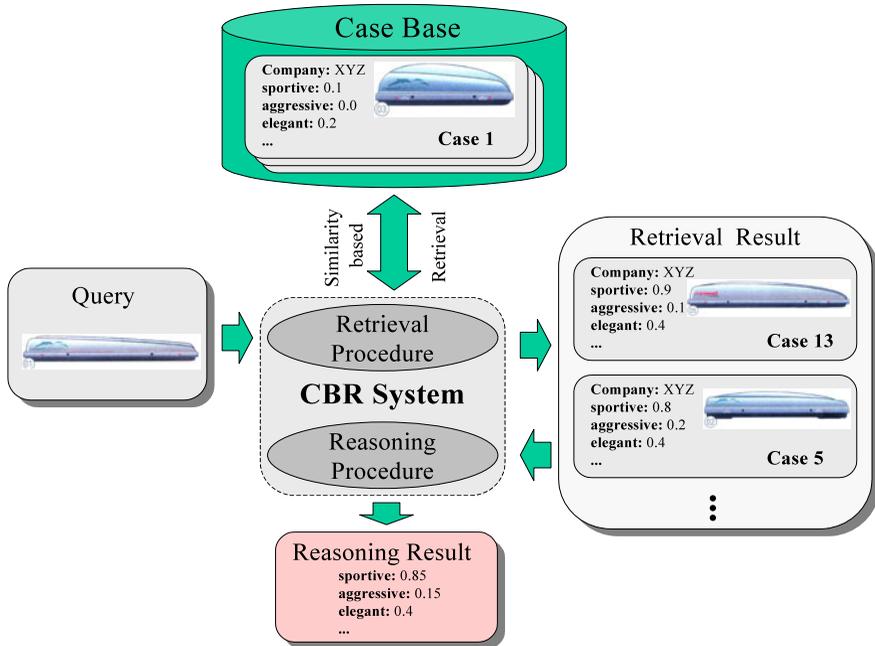


Fig. 18.14 Determining the aesthetic character

18.7 Knowledge Containers

Next, we discuss the influence of the above considerations on the structure and content of the knowledge containers.

The general situation is complicated when images are involved. Additional knowledge is needed to understand the query, the solution, and to support retrieval. The knowledge base contains two kinds of documents that can be called.

Similarities can also be defined between image objects from different levels. There are two elements to be considered:

- (a) Image processing methods.
- (b) Definitions of concepts and descriptions of how they are used.

The image processing methods compute attribute-values like texture or density and detect and describe geometric objects. These methods mainly connect the pixel level and the geometric level but have some influence on the symbolic level too. Hence these methods form a case base in their own right.

If the virtual phone conversation (described in Sect. 18.5.3) is performed between a human and a machine or between two programs within a machine, several such methods have to be called. The advantage over answers given by humans is that the information obtained from processing methods is more precise. This is relevant for diagnostic purposes and the therapy, for instance, degrees of medication.

The concepts that connect the geometric and the domain levels are used for the overall level as well. The description of the geometric objects is connected with the image processing methods that detect or generate them. Finally, the relation to the overall level has to be captured. Such knowledge allows using images and cases for application purposes.

In the approach described so far it is assumed that the knowledge is stored in additional case bases. This has the advantage that access to the knowledge is clearly organised by submitting queries to the bases. The way these queries are propagated can be organised in an influence diagram.

18.7.1 *The Vocabulary Container*

For the vocabulary there are no specific requirements. In general, the same comments as those for speech are applicable.

18.7.2 *The Similarity Container*

Here the level structure will play a major role. It is to a large degree responsible for the heterogeneous character of the representation form and will influence the similarity measures heavily.

The global measure operates on the overall level. These measures are of application-dependent character and are not different from the situations considered in the previous chapters. We have next a brief look at all of the levels.

18.7.2.1 Pixel Level

Here, the attributes are pixels; the domains are the grey or colour values. No measure with a symbolic meaning can be defined here. However, for comparing images there are some useful measures. In addition, on the pixel level we have already defined feature values such as centre of gravity or grey value; these can be useful for comparing images.

18.7.2.2 Geometric Level

Attributes are geometric entities with shape, contour, texture, and size. The values are computed by image processing methods. They are often of fuzzy character and the similarity measures have to respect this. Very often, weighted Euclidean measures are used.

18.7.2.3 Symbolic and Domain-Specific Level

On this level certain combinations of geometric objects are considered. The attributes refer to the geometric level but are considered from the domain point of view. The attribute-values are often symbolic, e.g., $\text{dom}(\text{size}) = \{\text{small, medium, large}\}$.

If the graph representation is taken, one can use the graph similarities described in Chap. 5, Case Representations. There one finds many possibilities.

The same works for sketch-based modelling, mentioned earlier. The graph similarities allow comparing skeletons as complex mathematical objects.

18.7.2.4 Overall Level

On this level one finds objects from the symbolic and the domain-specific levels. Here, understanding takes place. Specific attributes of interest are used on this level. There are, for instance, relations between components like relative, “relative position”, and “absolute position” or certain attribute-values from lower levels.

The measures on the lower levels are the local measures. On the pixel and the geometric levels there are mathematically defined measures. If the objects shown in the image do not have a fixed position, a demand on the measure is that it be transformation-invariant for translations and rotations.

On the domain-specific level the basic (atomic) objects are those components that are not decomposed anymore on this level. Examples of such atomic components could be the different ventricles in the medical example.

These attributes and predicates are associated with objects that allow describing the image in a component-oriented way according to the local-global principle.

On the geometric level the approximation by polygons plays a particular role, for instance, in showing which deformations can be tolerated and which cannot. As a consequence, similarity measures can be defined for comparing them.

The local similarities carry much knowledge. The weights reflect the importance of an attribute for the intended purpose. This heavily uses knowledge of the intended domain, as in medicine: “Left oval and right oval are in normal situations symmetric to some axis but not necessarily if pathologies are present”.

A specific problem is that images are compared with text and/or symbolic descriptions. For this purpose one needs access to the case bases on the domain-specific level.

18.7.3 The Case Base Container

For cases there are different possibilities:

- (a) The query is a symbolic description and the answer is an image.
- (b) The query is an image and the answer is a symbolic description.
- (c) Both the answer and the query are images.
- (d) The query and/or the answer are skeletons. This plays a role in classification, where the answers are classified skeletons and the query asks which class does a given skeleton belong.

Most of the tasks can be regarded as classification problems and we will mainly refer to them. Unless one has a catalogue of precisely defined objects, not all possible images will be stored. That would in fact be impossible because there is an infinity of vaguely or informally defined variations of interesting objects. To deal with this problem one can use fuzzy representations.

The situation is different if the solution is an action. Then an action may contain a parameter that occurs in the image description as well as in the therapy. The consequence is to extract the parameter value from the image as well as possible.

Here one uses the concept of a category. A category is something like a set but it has no crisp definition; it could be a fuzzy set. Instead of a definition, one has typical elements called prototypes. For these objects the community in question agrees whether or not the object certainly belong to the category.

Examples in medicine are:

- prototypical images of non-pathological objects,
- prototypical images of certain tumours.

It is clear that a category may have more than one prototype and the prototypes may not even be similar to each other. For instance, in medicine there may be different kinds of appearances of tumours of the same origin.

There are essentially two different kinds of case bases. The first kind is not collected for special CBR purposes. The collection was done in an institution or company in a certain domain for various reasons, as with a collection of images in a hospital. A CBR system can make use of it by retrieving an image when needed.

The second kind consists of systematically collected cases for CBR purposes. This is the case when one is interested in a specific class of problems and is collecting prototypes.

The first kind of collection is called an archive. The organisation of the archive is done as in a database. Each image has an index that allows a database retrieval.

Examples:

- (a) For air views this can be the geographical location.
- (b) In medicine this is mostly the name of the patient and the date when the image was taken.

This is just document retrieval and one does not have to look at the image itself. The problem arises when one is searching for images where these kinds of indices alone do not help.

Examples:

- (a) Search for such images where the houses have a swimming pool in the backyard.
- (b) Search for images of knee topographies with a certain property of medical interest.

In these examples one searches for an object on the domain-specific level. For this purpose the images themselves have to be investigated. One way to do it is to present a prototype of what one wants to retrieve in the query. Of course, an index of the image can be of additional help.

If images are stored in a case base then it is useful to augment them by additional information like properties or explanations. Often, a query is lacking such information and a search of the case base returns an image with such information. This happens typically when the query is a skeleton.

If one stores images belonging to different classes one usually stores prototypes only. Given a query, the nearest neighbour is the most similar prototype. This is analogous to the retrieval using Voronoi diagrams; see Chap. 8, Retrieval.

18.7.3.1 Retrieval Aspects

Many retrieval methods shown in Chap. 8, Retrieval, do not work for images because they refer to specific case representations. The standard problem of indexing cases is particularly problematic for images. It is still investigated intensively. A major technique is to include metadata in the index. Metadata is defined as data providing information about one or more aspects of the image data, such as:

- Means of creation of the data.
- Purpose of the data.
- Standards used.

In fact, standards have been introduced for this purpose. However, many collections of images are not indexed this way. In principle we have the same two views for retrieval from the case base as in text retrieval:

- (a) Document-oriented view: We search for images as documents. This requires indexing. If indexing is present then the retrieval is not different from any other document search and the usual methods for information retrieval can be used. In particular, no understanding of the image is required. An example is the retrieval of X-rays of some patient.
- (b) A variation is the matching view: Two images are matched on the pixel level (with some metadata used). This is the pattern recognition view, and again the meaning of the image is not considered.
- (c) The content-oriented view: We have to understand images. An example is that of searching for an X-ray image of a leg broken in an unusual way. For this we need different levels of representation.

The starting point of the retrieval is a query. In a summary of queries we see five major types:

- (a) Show me an image that meets a description on the overall level.
- (b) Show me an image that contains an object on the domain level with certain properties.
- (c) For this query image show me the most similar one with respect to the disease in question from the case base.
- (d) What properties do some objects on the image have?
- (e) What properties does the whole image have, for instance, to which class does it belong?

The domain-specific level is central for the queries (b), (c), and (d). The overall level is needed for the queries (a) and (e).

For organising the retrieval, we follow the analysis given above. The first difficulty is that often the attribute-values are difficult to compute. These values can be computed at design time or at runtime. In the first situation it seems that one can omit the image itself and replace it by a symbolic description. This neglects some aspects:

- (1) It may be required to measure and compute certain properties of the image that are not computed a priori. For instance, one cannot compute the size of all objects in the image in advance.
- (2) The uncertainty in the figures in the image require us to investigate them individually.
- (3) It may be too costly to compute the attribute-values a priori because only a few of them are used in a specific application.

In order to investigate the image at runtime one has to use the level hierarchy. Depending on the query this can be done bottom-up as well as top-down. The corresponding steps require calling case bases from the knowledge base (coming from the experience factory).

18.7.4 The Adaptation Container

When an image is presented as a query, image processing methods are often used for adapting the image for better understanding. Examples are enhancing of contrast or segmentation or the methods shown above for the ventricle. If the query is symbolic then the methods of Chap. 9, Adaptation, apply.

When an image is in the solution, there may be a need for adaptation too. The images are stored in the case base but they may not be totally satisfactory. Often, a useful adaptation is that of marking regions of interest. Other examples that occur frequently are when images are shown to customers. An example is:

A store is selling houses in a vacation village in another country and typical images of the houses are in a catalogue. When the houses are modified the images have to be adapted too.

The use of rules for describing adaptations is needed, for instance, to modify images. Therefore, a different approach is used. This means:

- (1) First a query for an adaptation is formulated.
- (2) Then this query is addressed to the image processing case base.
- (3) The retrieved result is applied to the image in the query.

The procedure allows formulating the reuse phase in this context. The essential difference is that images should be adapted.

18.8 Revision

In all applications the system has to be tested and revised. The revision will not change or improve images but may replace images by other ones or add new ones. In particular, in medical applications the significance of images has to be updated continuously. This applies also to transfers from one hospital to another, in particular, if the new hospital is in another part of the world. All this is the purpose of maintenance.

18.9 Standards

In several application domains there are standards for recording and storing data that CBR systems should recognise. In medicine, DICOM (Digital Imaging and Communication in Medicine) is a standard for handling, storing, printing, and transmitting information in medical imaging. It includes a file format definition and a net-

work communications protocol. The standard deals with the exchange of digital information and allows sharing radiological images across different locations. It makes it possible for physicians to interact and exchange analyses through the Internet using some protocol.

A major use is image procedure management, in particular, for patient records. It may also contain embedded references to other structured reports, images, waveforms, and audio. It encodes essentially semantic information. For this purpose, DICOM draws up a real-world ontology.

18.10 How to Design Such a CBR System?

The general principle for designing a system was already discussed, and we will comment on some image-specific issues only. As for each design of a CBR system, one has to fill the containers and specify the methods in the process model.

For building the image processing base, there are sufficiently rich and well-organised libraries available. The geometric case base is a little more specific. However, many geometric objects are of interest for different applications too. It is application-dependent the decision of which geometric objects should be stored.

A crucial point when considering similarity measures is the gap existing between traditional attribute-value representations and the representations used for images. For establishing similarity measures one should proceed as follows:

- (1) Describe the images in text as well as possible.
- (2) Describe in text what properties one finds relevant and characteristic.
- (3) Relate these properties to properties that can be defined in terms of images.
- (4) Rank these properties according to importance.
- (5) Compress the properties into attribute descriptions.

For these tasks one has to build a knowledge base. If one has established this base one can proceed as follows:

- (a) Investigate which elements can be stored in the knowledge containers and determine how to address the knowledge base.

This supports bridging the gap between descriptions. In the sense of Chap. 3, Extended View, one has to define a mediator language for building the bridge.

18.11 Applications

There are several application areas, from aerial photography to medicine. The boundary with pattern recognition is not strict but we do not discuss this here; see Chap. 23, Relations and Comparisons with Other Techniques. We limit the scope here mainly to applications in medicine. Medicine is an important domain for any application of imaging techniques. The images investigated are usually X-rays or

topographies. We encounter both document-oriented and content-oriented retrieval; for the former, the consideration of the lower levels is often sufficient.

There are huge archives that could be used as case bases. The problem with archives in hospitals is that they are mostly not topic-oriented but, rather, patient-oriented. Therefore, it is necessary to investigate the images themselves and to interpret them during the search. In radiological clinics, much effort and time of humans is spent on this. The search for images with respect to content orientation should make use of standards like DICOM.

There are two potential ways to organise medical images:

- (a) A case base including healthy and sick patients. Healthy patients would be called the positive examples, while sick patients would be the negative examples.
- (b) A case base of sick patients with diseases associated with one part of the body only (e.g., brain, liver, heart). In this way, other case bases could be built for other parts.

The use of the first type of case base is to detect possible pathological elements. This suggests considering two areas of the rough set method: The surely pathological area and the uncertainty area. Both are of interest.

Other applications look at identifying classified objects on images like cars in traffic situations. The purpose is to identify certain cars. If witnesses describe a car they often draw a rough sketch manually. For getting more precise information one can apply similarities between skeletons and documented images in a case base.

Another application field concerns the use of spatial information from surveying, remote sensing, and aerial photography. This kind of application requires special spatial problem-solving techniques and spatial similarity measures.

18.12 Tools

There are libraries for image processing methods. One example is Matlab (<http://www.mathworks.com/products/matlab>), another one is Smartcode (<http://image-processing-tools.smartcode.com>).

For image retrieval there are many tools available, mostly for special purposes. Some examples of general content-oriented commercial systems are IBM's QBIC (<http://www.research.ibm.com/topics/popups/deep/manage/html/qbic.html>), and Excalibur's Image RetrievalWare; see Eakins and Graham (1999).

The FM Ultranet tool (Balaa and Traphöner 2003) is used in the domain of ultrasonography for detecting abnormalities in fetuses. It is built on top of CBRWorks (<http://cbr-works.net>).

18.13 Chapter Summary

This chapter is concerned with images in the CBR context. First, the representation of the objects was introduced. Following the local-global principle, images have

been considered from the view of a level structure. This starts from the elementary pixel level and goes up to the geometric and the domain levels and reaches finally the highest level of abstraction, where the meaning of the whole image is considered. This is a holistic view and does not consider the higher abstract levels only.

The storage is in various case bases located at the different levels. On each level we considered representations, methods, and similarity measures. A useful form of representation on the domain level is graphs.

The semantics is represented as a semantic function using the local-global principle too. According to this principle the similarity measures are obtained in the same way. A main interest is in image understanding because this is needed when images are queries or answers. As a special kind of semantics, aesthetics was investigated.

We also distinguished whether the image occurs in the query or the solution or in both. Relations to image retrieval are discussed. This plays a role in archives and is related to standards like DICOM.

18.14 Background Information

Document-oriented retrieval and image matching can be done with or without image understanding. Most applications of matching that are related to pattern recognition require no or almost no understanding of the meaning of the image. Examples are the matching of faces and fingerprints. As with pattern matching, CBR has no significant contributions in this area. The relation of CBR to pattern recognition and information retrieval is discussed in Chap. 23, Relations and Comparisons with Other Techniques. The role of the influence potential is detailed in Li (1993).

CBR requires image understanding but cannot be separated from the lower levels. Many applications of CBR to problems that concern images (with some emphasis on the medical domain) can be found in Perner (2007). In Perner (2001), a level approach similar to the one developed in this chapter is explained.

For indexing images, standards have been introduced; see IPTC (2007). This document specifies metadata properties intended to be used primarily but not exclusively with photos. An application of CBR to image segmentation is given in Frucci et al. (2008).

Applications of histograms and grey scales to image analysis can be found in Haralick et al. (1973). For comparing sets A and B of points in the real plane, a common metric is the Hausdorff metric; see Hausdorff (1914). The Fréchet measure (Fréchet 1906) deals with curves and surfaces in a mathematical way.

In human image understanding, the holistic view has a long tradition. However, humans have more complex “elementary” images that they consider as “basic”, e.g., “glasses”. These images are on an abstract level not anymore decomposed. Such basic images have been identified in art for describing pictures of Picasso or ornaments from ancient Greece. This gives rise to other types of “syntax of visual information”. In art this is quite popular; see Pettersson (1993). In Dondis (1973) the concept of visual literacy is developed. This says that understanding images is based on being able to view images in the light of a visual language. The geometric elements are

the raw materials for other levels of visual intelligence. However, the approaches for using these insights in computers have been rather limited. These approaches are in the same spirit as the holistic, level-oriented view developed in this chapter.

Sketch-based modelling is widely used in computer graphics, for instance, to improve interfaces to programs; see, for instance, Igarashi et al. (1999). One of the many applications is time series analysis where the series is represented as a stroke. CBR techniques were applied to sketch-based retrieval of architectural design in Gross et al. (1994). An overview of sketch-based modelling is given by Olsen et al. (2009). These techniques are closely related to the polygon approaches in this chapter. The term Content-Based Image Retrieval (CBIR) seems to have originated in 1992, when it was used by T. Kato to describe experiments in automatic retrieval of images from a database, based on the colours and shapes present; see Kato (1992).

The study of aesthetics was the subject of the European project Fiore; see Fiore (2013), from which we took Figs. 18.13 and 18.14. Several systems have been developed that apply CBR for image retrieval and interpretation at the symbolic level. Usually the images are not processed, and the symbolic terms are user-specified.

Early examples of retrieval of radiological images include Macura and Macura (1995), and Haddad et al. (1997) for the detection of coronary heart disease. Jaulent et al. (1998) retrieve images for the diagnosis of breast cancer in histopathology.

The use of images in CBR for medical diagnosis based on abdominal CT images can be seen in Grimnes and Aamodt (1996). They use a two-level structure. One is concerned with image processing with emphasis on segmentation. The second one deals with the overall medical structure. For a real-world ontology, see Revet (1997). A language for describing radiological images can be found in Abdala et al. (2003).

The DICOM standard is developed by the American College of Radiology and the National Electric Manufacturers Association. For an introduction see Pianykh (2008). Several attempts are going in the direction of using case bases too. A specific support of CBR can enable a user to find lists of images or other such objects and then retrieve them.

Several of the bottom-up ideas go back to von Wangenheim (1996). A comprehensive approach to image understanding and retrieval is in Cyclops (2013). There, the bottom-up approach to understanding is also utilized.

Spatial images have played a role in CBR several times. Research described using spatial similarity measures for comparing the location of objects in aerial photos is given in Carswell et al. (2002). Holt and Benwell (1996) reported the first application of CBR to geographic information systems (GISs). Their approach consisted of combining CBR with GIS to form a hybrid system to solve spatial problems.

In Weber-Lee et al. (1996) one finds a method to compose a solution for reuse as the most typical from a set of retrieved instances by applying typicality theory.

18.15 Exercises

Exercise 1 Take an interesting bottle of your choice. Describe the characteristics of its shape using a polygon approximation and the turns.

Exercise 2 How would you describe the object “swimming pool” in an image taken from the air?

Exercise 3 Describe your house in a graph representation.

Exercise 4 Define adaptation rules that allow changing domain-specific image objects by applying

- (a) Rotation
- (b) Linear Transformation.

References

- Abdala DD, Richter MM, dos Santos T et al (2003) CycML—a language to describe radiological images. In: CBMS 2003: 16th IEEE symposium on computer-based medical systems. IEEE, Los Alamitos, p 145
- Balaa ZE, Traphöner R (2003) Case-based decision support and experience management for ultrasonography. In: Reimer U, Abecker A, Staab S, Stumme G (eds) Wissensmanagement 2003: Professionelles Wissensmanagement, Erfahrungen und Visionen, Luzern, 2–4 April 2003. LNI, vol 28. GI, Bonn, p 277
- Carswell JD, Wilson DC, Bertolotto M (2002) Digital image similarity for geo-spatial knowledge management. In: Craw S, Preece A (eds) ECCBR 2002: advances in case-based reasoning. 6th European conference, Aberdeen, UK, September 2002. Lecture notes in computer science (lecture notes in artificial intelligence), vol 2416. Springer, Berlin, p 58
- Cyclops (2013) The Cyclops group. <http://www.cyclops.ufsc.br>. Accessed 28 Feb 2013
- Dondis DA (1973) A primer of visual literacy. MIT Press, Cambridge
- Eakins JP, Graham M (1999) Content-based image retrieval. Technical report 39, JIST technology applications programme. JIST, London
- Fiores (2013) <http://www.fiores.com>. Accessed 28 Feb 2013
- Fréchet MR (1906) Sur quelques points du calcul fonctionnel. Dissertation, École Normale Supérieure Paris
- Frucci M, Perner P, Di Baja GS (2008) Case-based reasoning for image segmentation. *Int J Pattern Recognit Artif Intell* 22(5):829–842
- Grimnes M, Aamodt A (1996) A two layer case-based reasoning architecture for medical image understanding. In: Smith I, Faltings B (eds) EWCBR-96: advances in case-based reasoning. Third European workshop, Lausanne, Switzerland, November 1996. Lecture notes in computer science (lecture notes in artificial intelligence), vol 1168. Springer, Berlin, p 164
- Gross M, Zimring C, Do E (1994) Using diagrams to access a case base of architectural designs. In: Riitahuhta A, Sudweeks F (eds) International conference on artificial intelligence in design. Springer, Netherlands, p 129
- Haddad M, Adlassnig K-P, Porenta G (1997) Feasibility analysis of a case-based reasoning system for automated detection of coronary heart disease from myocardial scintigrams. *Artif Intell Med* 9:61–78
- Haralick RM, Shanmugam K, Dinstein I (1973) Textural features for image classification. *IEEE Trans Syst Man Cybern* 3(6):610–621
- Hausdorff F (1914) Grundzüge der Mengenlehre. Reprint, Chelsea Publishing Company, New York, 1978
- Holt A, Benwell GL (1996) Case-based reasoning and spatial analysis. *J Urban Reg Inf Syst Assoc* 8(1):27–36

- Igarashi T, Matsuoka S, Tanaka H (1999) Teddy: a sketching interface for 3D freeform design. In: 26th annual conference on computer graphics and interactive techniques. ACM SIGGRAPH'99, Los Angeles, CA. ACM, New York, p 409
- IPTC (2007) Photo metadata white paper. IPTC standards. Document revision 11. http://www.iptc.org/std/photometadata/0.0/documentation/IPTC-PhotoMetadataWhitePaper2007_11.pdf. Accessed 28 Feb 2013
- Jalut MC, Le Bozec C, Zapletal E, Degoulet P (1998) Case-based diagnosis in histopathology of breast tumours. *MedInfo* 9:544–548
- Kato T (1992) Database architecture for content-based image retrieval. In: Jamberdino AA, Niblack CW (eds) *SPIE/IS&T 1992: image storage and retrieval systems*, vol 1662. SPIE, Bellingham, p 112
- Li X (1993) Potential analysis for massively parallel computing and its application to neural networks. Dissertation, University of Kaiserslautern
- Macura R, Macura K (1995) MacRad: radiology image resource with a case-based retrieval system. In: Veloso MM, Aamodt A (eds) *ICCB-95: case-based reasoning research and development*. First international conference on case-based reasoning, Sesimbra, Portugal, October 1995. *Lecture notes in computer science (lecture notes in artificial intelligence)*, vol 1010. Springer, Berlin, p 43
- Olsen L, Samavati FS, Costa Sousa M, Jorge JA (2009) Sketch-based modeling: a survey. *Comput Graph* 33:85–103
- Perner P (2001) Why case-based reasoning is attractive for image interpretation. In: Aha DW, Watson ID (eds) *ICCB-2001: CASE-based reasoning research and development*. 4th international conference on case-based reasoning, Vancouver, BC, Canada, July/August 2001. *Lecture notes in computer science (lecture notes in artificial intelligence)*, vol 2080. Springer, Berlin, p 27
- Perner P (ed) (2007) *Case-based reasoning for signals and imaging*. Springer, Berlin
- Pettersson R (1993) *Visual information*, 2nd edn. Educational Technology Publications, Englewood Cliffs
- Pianyk OS (2008) *Digital imaging and communications in medicine (DICOM): a practical introduction and survival guide*, 2nd edn. Springer, Berlin
- Revet B (1997) *DICOM cook book for implementations in modalities*. Philips Medical Systems, Eindhoven, Netherlands
- von Wangenheim A (1996) Ein konfigurationsbasierter Ansatz zur wissensbasierten Bildanalyse. Dissertation, University of Kaiserslautern
- Weber-Lee R, Barcia RM, Martins A, Pacheco RC (1996) Using typicality theory to select the best match. In: Smith I, Faltings B (eds) *EWCB-96: advances in case-based reasoning*. Third European workshop, November 1996. *Lecture notes in computer science (lecture notes in artificial intelligence)*, vol 1168. Springer, Berlin, p 445

Chapter 19

Sensor Data and Speech

19.1 About This Chapter

This advanced chapter is addressed to readers who work with and are interested in time series of sensor data and in the recognition of spoken language. We consider representations of cases and queries in such formats. We make use of signal processing but do not discuss detailed signal questions in speech. Rather, we are interested in problems concerning use of CBR, such as using signals as representation languages for cases. As far as speech is concerned, we also look at is-a-relation with conversational CBR (see Chap. 20, Conversational CBR). General knowledge about Part I is required and a look at Chaps. 17 and 18 is recommended.

19.2 General Aspects

Sensor data in time series play a major role in many applications. They are generalizations of static observations and often crucial for the diagnosis of faults and diseases. In medicine as well as in engineering, a diagnosis quite often depends on the temporal development of observable magnitudes.

Speech by its very nature falls into this category because it produces signal data and happens in a temporal development too. Recorded speech is in the first place a time series of signals. This time series has to be recognised properly. In a clean environment or when a headset is present, there are excellent recognition tools commercially available. We are concerned with situations where this is not the case.

In the local-global view, we distinguish again different levels of abstraction. The levels are quite parallel to those in image recognition. On a higher level of abstraction, humans with respect to a certain intention and context can understand sensor data and speech signals. Humans are sometimes able to perform the understanding. In many situations the data series do not, however, give direct insight into the underlying reasons. For this purpose, computers have to be programmed.

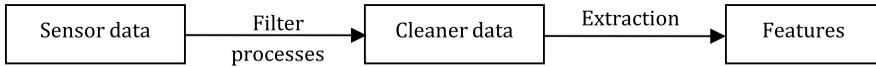


Fig. 19.1 Feature extraction

In a very general sense, we look at signal sequences and utterances as language elements. Of course, they are different from what we consider usually as human languages.

Examples are:

- If a medical doctor is listening to a cough, it is understood to be for making a diagnosis.
- If a car makes noise, mechanics know how to interpret it appropriately.

There is a basic difference between recorded speech and recorded measurements: The speaker has an intention for sending a message. Therefore, the proper receipt of the message can be controlled a posteriori. When measurements are recorded, one gets a message too. However, it is decided by the user what kind of message this is. For instance, if the measurements control a machine, one can say “the machine speaks to you”. Machine experts can understand this “language” and it is the purpose of automated methods to “learn” this language. The success of learning is decided by the correctness of the conclusions based on it.

These messages can be processed in different ways. When using CBR one proceeds in principle as follows:

“Last time the meaning of a similar message from the machine was M ; therefore, it is likely that the meaning of the message is now similar to M ”. The time series obtained can be of arbitrary complexity. In addition, the processes generating the data can be of a stochastic nature with probabilities that are unknown.

Mostly, humans interpret time series subjectively. However, humans have problems extracting the knowledge hidden in a time series. Often, they analyse data series manually, which is time consuming. However, they are better at understanding spoken language. The examples shown above illustrate this.

Humans rely on capabilities of the human ear or the human eye in order to grasp a scene as a whole and interpret it accordingly. The human may get an immediate impression when looking at recorded data. When using automated methods one cannot make use of such mechanisms unless one has established a suitable program. This is discussed in Chap. 18, Images.

We restrict ourselves to one-dimensional data. This is another difference with images where such a restriction makes no sense. The data are defined by a function $S(t)$, ranging over time t . In order to apply methods of AI in general and of CBR, in particular, the given time series has to be transformed and simplified, usually several times. At the end one wants to get a small series of numbers called features that are collected as feature vectors. These features are nothing more than attributes, and on this level similarity measures operate as usual. Figure 19.1 shows the main transformations in a simplified way.

For proper analysis the measurements are broken into segments that are also called windows. Filter processes clean these segments.

The final results of these transformations are the features. The goal is that they be simple but still convey significant information to human experts. Because most of the information is related to some kind of classification, we concentrate the discussion on this view. A very general structure of a system receiving and using measurement data for further processing is discussed next.

The steps are:

- (1) Removing noise.
- (2) Breaking the signal sequence into segments.
- (3) Extracting the features using extraction methods: The features are the only elements that are furthermore considered.
- (4) Classifying and identifying the features as the central step towards understanding. This step uses a case base that in this situation is a set of recorded signal sequences. It is the first place where CBR enters into the discussion.
- (5) Testing the system.

The first three steps perform preprocessing. The classification and identification step is where CBR and similarity measures will enter the process.

The test step is accompanied by revision; it is the revise step in the CBR cycle. Of course, the technical and mathematical details are quite involved and require some knowledge about the mathematical background. We try to discuss this briefly because the reader will not build such systems from scratch. On the other hand, even proper use will necessitate some knowledge about the underlying difficulties.

Next, we first consider general aspects for measured data. Then we analyse the special problems with and techniques for speech recognition.

19.3 The Level Structure

The analysis of sensor data and images has in common that they all start from very elementary elements as signals or pixels. These elements do not have a meaning attached to them. One can obtain a meaning only at higher levels of abstraction when certain structures are revealed.

The approach is in principle the same as that for images, namely, to introduce a level structure starting from the signals and reaching the overall understanding at the highest level. In presentations as well as in the similarity measures we follow this structure. It requires a deeper investigation of how the investigated objects are presented for similarity measures.

For sensor data the lowest level contains the signals provided by (possibly) stochastic processes. These data may be noisy, due to external or internal noise.

For reaching the level that allows extracting features one starts with the signals as the very low-level elements.

If we start from stochastic processes, we restrict the discussion to hidden Markov processes (HMMs). Therefore, the approaches to these processes and dynamic situations are of interest. Stochastic processes have been considered in Chap. 16, Probabilities.

In general, we have the following levels:

- (1) The level of the stochastic processes, HMMs: This is concerned with signals.
- (2) The level of the features: Here, the signal sequence is represented formally and in a way that allows comparisons by similarity measures.
- (3) The language level. Here, combinations of features are formed that are of interest in the applications and can be denoted by names. This is a level of symbolic representation.
- (4) The overall level: Here, the meaning of the signal sequence is understood.

19.3.1 The Low Level of Signals

On this level, several transformations take place that are considered as preprocessing steps.

- (1) If the data are recorded as an analogue signal $S(t)$ then this has to be transformed into a discrete time signal sequence $S(n)$ of for further processing. Discrete time means that one can represent the sequence on integer-based indices. The transformation is done by collecting the values (samples) at discrete times only.
- (2) The next step is breaking the signal sequence into segments that are easier to handle. This is of particular importance in speech recognition.
- (3) Another widely used step changes the representation of the signals. Besides using the time domain, one can represent the signals in the frequency domain. In the time domain one can see how a signal changes over time, whereas in the frequency domain one can find the frequency information of the signal. It shows how much of the signal lies within each given frequency band over a range of frequencies. This is somewhat analogous to histograms for images.

On the frequency level, filtering may take place too. Under filtering we subsume a number of individual processes.

A major part of filtering is to remove noise. There are two origins of noises.

- (a) One comes from the imprecision of noise generators or the imprecision of sensors.
- (b) The other comes from additional noise generated by the environment.

To judge the imprecision of the noise generators, one needs knowledge about their structures. This is the subject of control engineering, and will not be discussed here in general. In the case of speech we provide some more information below.

This level is after preprocessing and also the starting point for feature extraction. This process connects the low level and the feature level.

19.3.2 The Feature Level

The first difficulty is that one has to start from a low level and it is not immediately clear how the features should be defined. The reason is that the features have to be defined on a low level but they are interpreted on a higher level. Therefore, one employs general mathematical models on the lower levels for feature extraction. The principle is that one can reconstruct most of the original sequence from the features.

Feature extraction means identifying certain attributes of interest. The feature extraction is usually a non-invertible transformation that loses some information.

The purposes are:

- Capture essential information of measurements.
- Compress information into a manageable form.

After extraction these vectors should still contain sufficient information for characterising the original message. For this reason, the feature vectors should facilitate case indexing.

The features are the objects that represent symbols and are compared by similarity measures. Feature vectors can be in queries as well as in answers. In general, this is not new and is used in CBR throughout. In many situations where CBR is applied, humans have defined features (i.e., attributes) before becoming interested in CBR, and therefore the extraction step does not have to be repeated. For this reason the feature extraction is not named in the CBR cycle; it is, rather, hidden in the problem formulation step. In general, there is no systematic method yet developed to extract attributes from a problem formulation. What has been done is research on the importance of the attributes; see Chap. 6, Basic Similarity Topics, where weights are discussed.

Generally in CBR, the understanding of the feature vectors requires knowledge that can be partially attached to the case. There are different kinds of features in data recognition but they share some commonalities.

Initially the signal sequence is recorded in the time domain. For defining features that contain information, one switches to the frequency domain. Now the time axis is replaced by the frequency axis. An initial problem is to define what the features for a time sequence should be. Today, one distinguishes mainly two types of features.

The first one is of statistical nature. The extracted features such as mean, variance or correlation coefficients are sometimes called metadata in statistics. They are discussed in Chap. 16, Probabilities.

Another principal and frequently used kind of method uses representations of $S(n)$ in terms of certain basic functions ϕ_i . Here we assume that the signal already occurs in discrete times $n = 0, 1, 2, \dots$

The representation is

$$S(n) = \sum_i c_i \phi_i(n).$$

Here, the coefficients c_i (or the first ones) serve as features.

A widely used instance of such representations uses trigonometric functions. The most elementary situation is the representation as a Fourier series. In this case the

transformation is called the Discrete Fourier Transform (DFT) of the signal. It is an example of transforming the time domain into the frequency domain. It decomposes a function into the sum of sine wave frequency components.

A very popular and commonly used method for feature extraction is the Linear Predictive Coefficient Method (LPC). This is a method for predicting the values with a high accuracy based on the past and therefore on the known values. It relies on the tacit assumption that the values are not completely independent of each other, a situation that often occurs in practice. In many applications such as in speech, there is some evidence for this assumption.

A first problem is that the coefficients are not known. Even in the best case this prediction is not totally exact and therefore the representation is equipped with an error. The signal S is modelled as a linear combination of some previous samples:

$$S(n) = - \sum_{i=1}^{NLP} a_{LP}(i)S(n-i) + e(n).$$

Here, N_{LP} is the number of coefficients, a_{LP} is the linear prediction coefficient and $e(n)$ is an error term. It is the difference between the predicted value and the actual measured value. The features are the prediction coefficients. In general, the coefficients are unknown and have to be estimated. The goal is to minimize the expected error. There are various ways to do this and they rely in general on statistical elements when the stochastic process is observed.

This can be improved by splitting up the domain of the original signal sequence $S(n)$ into segments (intervals) I_k , which gives rise to two indices, one for the basic functions and k for the intervals:

$$S(n) = \sum_{i,k} c_i^k \phi_i(n).$$

The two indices i and k contain more information because they refer also to the intervals. From interval to interval the characteristics of the process can change. In technical terms, we have a nonstationary process. This process is, however, almost stationary in small intervals but differs from interval to interval. The determination of the intervals is mathematically difficult. This is an example of minimizing errors, which rarely takes place in CBR.

19.3.3 The Symbolic and the Overall Level

The overall level is concerned with the goals intended by the measurements or the speech. For instance, this could be a fault diagnosis or some message during monitoring or conversation. On this level, reasoning takes place in the way humans do it and one does not refer to the lower levels anymore.

On the symbolic level, abstractions and combinations of features are located. Usually, they are predefined and arise from the application domain. These symbols

carry names and one can regard them as elementary parts of a symbolic language. In speech recognition they are words.

The symbolic and the overall levels are where conversational CBR takes place. One is not interested in the lower levels and the errors occurring there; for the user, one has completely eliminated them.

19.4 Semantics

The meaning of a recorded sequence from measurements is not immediately obvious. However, an experienced expert may know very fast what it means because of available background knowledge. As mentioned above, the recordings formulate certain properties in a “language” that one has to learn. This is a situation with which one is confronted in any knowledge-based system. The semantics of a recording is the meaning of the intended purpose. As for images, we denote the semantics by a semantic function SF:

$$\text{SF} : \text{recorded representation} \rightarrow \text{meaning.}$$

There are different levels of abstraction where these objects of interest can be located. Hence, it is natural that the meaning of an object be located on the next higher level of abstraction. We consider recorded representations on two levels: the feature level and the symbolic level.

A feature is a representation of a stochastic process that produces a signal sequence.

For the meaning of a general sequence of data there is no general rule. In speech this is easier because the meaning is the intended word or phrase. In general, we get the following for the meaning of the features:

$$\text{SF}_{\text{feat}} : \text{features vectors} \rightarrow \text{symbols.}$$

The features are recorded in the feature case base.

19.5 Remarks on CBR Process Steps

Next, we investigate the knowledge containers and the steps in the CBR process that are relevant for this chapter. As much as possible this is done for both measured signals and speech.

19.5.1 Cases and Case Bases

The case base does not store the recorded measures directly. The stored objects are feature vectors representing the measures. The feature vectors can be compared with

a similarity measure. They denote actually occurring queries or solutions. Therefore, we can interpret the results of similarity as we always do in CBR: When the query or the problem is presented, then the nearest neighbour is supposed to provide the most satisfactory answer.

In case bases a case is a feature vector denoted by fv . It is obtained from a recorded measurement as described above. The solution set is the case base

$$CB = \{fv_i | i \in I\},$$

where I is some index set. A problem is now a feature too and the similarity measure is supposed to determine its nearest neighbour from CB .

Thus, the case base can also be regarded as some kind of product set in the extended view of CBR.

This view assumes that both queries and solutions contain such data. Often, one of them has another character. For instance, the query can be, “is this measurement pathological?” with the possible answers {yes, no}. However, there are two levels for representing cases: The symbolic level and the feature level. The symbolic level is accessible to humans. In fact, when humans describe cases they do it on this level. For this, a symbolic description is needed.

On the symbolic level synonyms frequently occur, in particular, in speech. In contrast to the textual situation, it is not sufficient to record them in a thesaurus because one cannot list all of them. In speech, synonyms can be uttered and be regarded as unknown words. Therefore, a case base has to contain all synonyms of interest. In addition, different pronunciations, as in dialects have to be recorded explicitly.

19.5.2 Similarity

Dynamic processes produce the sensor data, and the similarity measures have to be compatible; see Chap. 7, Complex Similarity Topics.

The definition of similarity measures takes place on the levels where one represents the objects. Therefore, we consider again the two levels:

- (a) The feature level.
- (b) The symbolic level.

As a consequence we will encounter two similarity measures:

- (c) The measurement similarity: On the feature level.
- (d) The symbolic similarity: On the symbolic level.

Measurement signals come in as a function $S(t)$ defined over time. The values are real numbers representing the signals. The first similarity measures operate on the feature level. They are responsible for a correct retrieval. These similarity measures sim_{feat} compare feature vectors fv :

$$\text{sim}_{\text{feat}}(fv_1, fv_2) \in [0, 1].$$

The most commonly used and a recommended measure is Dynamic Time Warping (DTW). This is described in Chap. 7, Complex Similarity Topics.

When measurements are considered on the symbolic level, one uses suitable expressions like “failure” or “no failure”. The symbolic similarity compares the stored and the uttered symbol from the viewpoint of the intended action. This approximates directly the intended utility. It is cost-sensitive because some errors do create various kinds of unwanted consequences. There is no difference with other symbolic representations or with the understanding of texts.

On the symbolic level, humans can decide about the correctness of provided information. This cannot be done directly on the feature level. What is wanted is that feature and symbolic similarity coincide or that at least the former implies the latter.

This gives rise to a kind of correctness demand for the relation between the measures: The symbolic similarity can be regarded as the specification for the feature similarities.

The symbolic similarity is related to semantic issues as discussed below. The feature similarities compare feature vectors that are invisible to the human but central for the retrieval of words that have a meaning and are central for message understanding.

19.5.3 Retrieval

For retrieval, most of the symbolic methods in Chap. 8, Retrieval, can be used only on the symbolic level, and that is not the only important level here. The feature vectors contain numerical data and the usable retrieval methods should tolerate them.

A question arises about the use of Voronoi diagrams in this context. At first this seems reasonable, but the problem arises when the vectors are fairly long, and this makes the diagrams not usually recommendable. For this purpose segmentation is done for reducing complexity.

19.6 Speech Recognition

In this section, the special aspects in speech recognition are discussed, and we rely on the concepts and techniques in the first part of this chapter.

Speech is a very difficult case of measured signals. Technical devices can generate it artificially, as in speech synthesis. The real challenge occurs, however, when speech is generated by humans in everyday situations and has to be received and understood there. That presents special difficulties to using machines for reception and understanding. It has been widely investigated and in this section we will have a look at it.

Despite the various applications of speech the recognition, it is itself an application of CBR, as we will point out. The case base contains examples of speech where the meaning has been determined in another way.

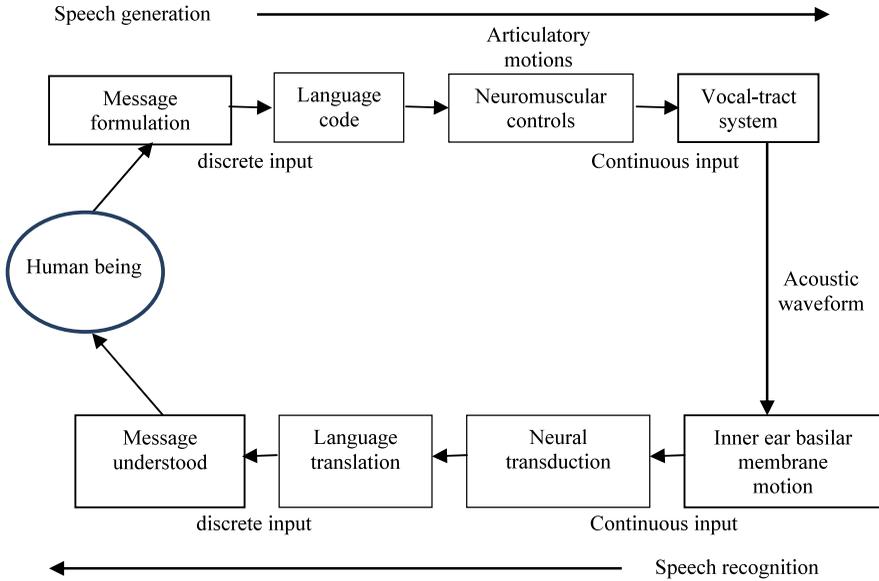


Fig. 19.2 Human speech generation and recognition

19.6.1 The Whole Speech Process

We start with describing the whole process from generating to recognising speech. This is necessary because humans are involved and one cannot neglect their intentions. The process sketched in Fig. 19.1 is now detailed.

It starts from speech intention and continues with speech utterance, speech receiving and finally speech understanding, as shown in Fig. 19.2. It extends the general process to the involvement of humans. The figure will only show what the human has to do but not how it is done. Machines will do it in different ways.

In Fig. 19.2, speaker and receiver are abstract agents. They can be humans as well as machines. First, we look at the situation where both are humans.

In the speech production shown in Fig. 19.2, first the message formulation takes place in the speaker’s brain and then language code is generated accordingly. Next, the neuromuscular movement takes place to control the vocal apparatus and to decide which organ to move: ears, or tongue or vocal tracts. Then the vocal tract system is used to articulate the message. The message is produced as a continuous speech waveform.

The message interpretation shown to the left in the bottom half of Fig. 19.2 is the speech recognition. This is done in several steps, of which one is a neural transduction of the spectral features into a set of sound features or distinctive features. These features are decoded and processed by the brain and the sound features are transformed into the set of phonemes, words and sentences in order to understand or recognise the message.

If one of the participants in Fig. 19.2 is a machine, the corresponding tasks have to be automated. Because speech is often embedded in a dialogue (as in conversational systems), speech synthesis is needed too. There are convenient available tools for synthesis and we do not consider it here further.

The most common and interesting situation is when the speaker is a human and the receiver is a machine. This takes place in Conversational CBR. The human speaker is not formally defined. In addition, different speakers will pronounce the same word in different ways. As we can see, the process has mental as well as physical parts that are located on different levels of abstraction. The message is created in the mind, where also the verbal formulation takes place. For imitating this on a computer, knowledge is needed. The speech articulation in the human body takes place in the vocal tract. It is done automatically by a “built-in system” in the body and the human does not have to care about it.

19.6.2 The Role of the Levels

On the process level, first an analogue function $S(t)$ is obtained, captured by a microphone. In the microphone the analogue function is transformed into a discrete function $S(n)$ defined on integers. These are the basic elements on which further operations take place.

The main part of receiving on the low level is identifying the words. The human is able to identify them even in unclear speech. The human has learned the words in question in the past from experience. That calls for using CBR here too.

In a computer, one needs for this purpose:

- (a) A collection of words or phrases that contain all possibilities that are intended to be spoken. This constitutes a case base.
- (b) A way to match an actually uttered phrase with a phrase in the case base. For this a similarity measure is required for performing a nearest neighbour search.

These steps constitute the CBR part of speech recognition. A spoken word presented in a query therefore gives rise to a search for the most similar one in a case base in order to detect it.

For understanding, one needs to grasp the meaning of the words on a higher level of abstraction. In any kind of understanding of messages, one has to refer to some context. In human understanding one can refer to higher levels of abstraction in a flexible way that allows the reference to contexts, as in text understanding.

Computers do not have this ability directly available; therefore, one needs for dealing with speech on the computer a holistic view integrating different levels. This is similar to the problem of dealing with images. Both images and speech start from a very elementary level, the pixels and the signals. For both, the meaning is of interest only on the highest level.

There are some essential differences. An image has a static representation; it is not changed or extended while one looks at it. Speech, however, is dynamically

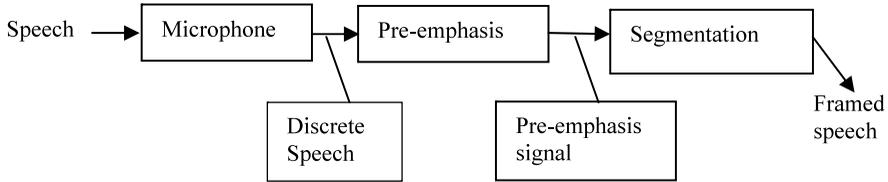


Fig. 19.3 Special aspects of an automatic speech recognition system

represented; it develops while one is listening. Hence speech recognition differs from other kinds of recognitions by its real-time character: One has to recognise immediately and has no time for extensive search.

19.6.3 Structure of a Speech Recognition System

The general structure of such systems is mostly the same for all technical realisations of speech recognition. They go bottom-up from the lower levels to the higher ones and follow roughly the approach in Fig. 19.2. The differences are on the left side, as shown in Fig. 19.3.

The interaction between the blocks is now:

- The microphone digitizes the input speech waveform.
- Pre-emphasis, namely, the pre-filtering, bridges the balance in the high and low frequencies.
- Breaking the signal into several segments is done in the segmentation block. These are in general termed windows, frames or blocks. These windows may overlap because the spoken signals are not independent of each other.

19.6.4 Cases and Case Bases

Cases occur on the lower level where they contain feature vectors and on the symbolic level where they contain words, phrases and sentences. We described the latter in Chap. 17, Textual CBR. In the next section we describe the relation between the features and the symbolic level.

19.6.5 Speech Feature Extraction and Similarity

In speech recognition, a vector obtained from an actually spoken utterance is the problem, and one wants to know which of the recorded sequences it is. After the utterance is recorded, the feature vector is extracted and compared with the vectors

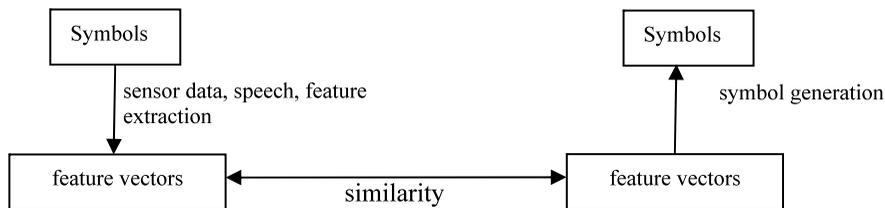


Fig. 19.4 The speech cycle

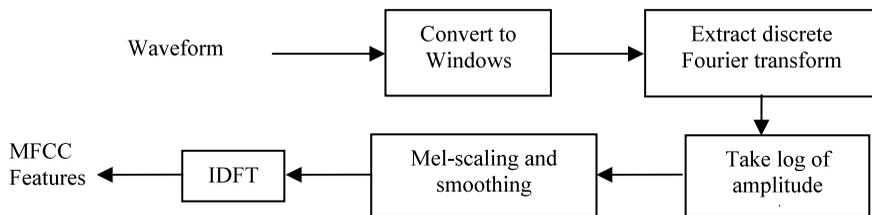


Fig. 19.5 Obtaining MFCC features

in the case base. There are different kinds of features in speech recognition but they share some commonalities. The feature level contains the features extracted from the speech. This is the level on which similarity measures and retrieval functions operate. From the features, in turn, the symbols can be generated. The CBR techniques take place on the feature level. This gives rise to a cycle illustrated in Fig. 19.4.

The left side describes spoken words and the right side refers to the case base.

First, the speech was recorded in the time domain. For defining features that contain more information, one switches to the frequency domain. Now the time axis is replaced by the frequency axis.

A more involved kind of coefficient that is special to speech recognition is the Mel-frequency Cepstral coefficient (MFCC). This analysis is based on the human perception methodology. It is observed that the human ear is a filter. It concentrates on certain frequency areas only. These areas are nonuniformly distributed on the frequency axis: More areas in the low frequency and fewer areas in high frequency regions.

This is the reason for commonly using the Mel-frequency Cepstral coefficients that are in the frequency domain. The purpose is to follow the perception of the human ear. Below, we show where they are used. We need the following terms:

- The “spectrum” of frequency components is the frequency domain representation of the signal.
- Mel-filters generate the mel-frequency cepstrum.
- The cepstrum is the computation of the logarithm of the absolute value of the output of the Mel-scale filter bank.

Figure 19.5 shows the main steps for getting the MFCC features.

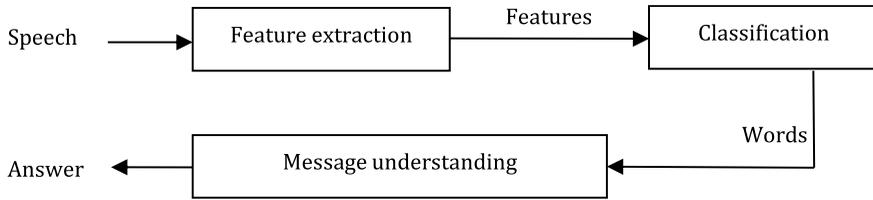


Fig. 19.6 From speech to understanding

We will not present the mathematical details, and mention the purpose of the steps only:

1. The input is in speech waveform and is preprocessed in the microphone.
2. The pre-emphasis emphasizes the frequency component by balancing the high and low frequencies.
3. Windowing (framing) is applied to obtain speech intervals.
4. Take the discrete Fourier transform (DFT) of the interval of the signal.
5. Take the logarithms of absolute values of output of the Mel-filter bank and obtain the cepstrum.
6. Take the inverse discrete Fourier transform (IDFT) of the cepstrum to obtain the MFCCs.

19.6.6 The Word Level and the Vocabulary in Speech

In speech, the measurement similarity is called phonetic similarity. The phonetic measure contains subjective elements insofar as it has to deal with subjective pronunciations and acceptance. It is located on a low level.

The symbolic similarity is on the word level. On the word level there are different kinds of subjective interpretations. For a spoken word, the nearest neighbour in phonetic similarity should be the word the speaker has intended to express.

The vocabulary is twofold. The vocabulary on the symbolic level is primary; it consists of the words and phrases that can be uttered. The vocabulary on the feature level is obtained by feature extraction.

19.6.7 The Overall Level

When a word or a phrase is obtained, a human connects a meaning with it. That is done by integrating it into the actual context. This leads to the representation of semantics that we see in Fig. 19.6.

It gives rise to a kind of correctness demand for the similarity measures. The symbolic similarity is related to semantic issues, as discussed next. The feature sim-

ilarities compare feature vectors that are invisible to the human but central for the retrieval with words that have a meaning, and central for message understanding.

The feature level is the one on which similarity measures and retrieval functions operate. From the features, in turn, the symbols can be generated. If a feature vector is obtained from a spoken word then the system searches for the nearest neighbour in the case base of features. If this denotes the intended word then the answer is correct. Here the human has to be involved because only the speaker knows the intended word.

19.6.8 Semantics

Speech and text have in common that they both deal with natural language and they both have to cope with a linear ordering. An essential difference is that text is a static object while speech is a process. A reader can study a text many times but it is often impossible to get repetitions of speech. For instance, in a telephone conversation one can just ask the partner to repeat the sentence.

For getting the meaning of sensor data there is no general rule. In speech, this is easier: The meaning is the intended word or phrase.

The features are recorded in the feature case base. Feature vectors f_v are generated from a symbol w . Therefore, we define

$$SF_{\text{feat}}(f_v) = w.$$

If in speech recognition a feature vector originates from a newly spoken word, there are two possibilities:

- (a) The word is not in the vocabulary: Then understanding is not possible.
- (b) The word is in the vocabulary: The nearest neighbour on the feature level determines the word.

Possibility (a) can be discovered only if the similarity to the nearest neighbour is below some threshold.

Situation (b) gives rise to the correctness concept of the retrieval: The retrieved word has to be the same as the uttered one.

19.6.9 Adaptation

In Chap. 9, Adaptation, two kinds of adaptation have been introduced, adaptation of the query and adaptation of the solution. In the context of this chapter, query adaptation takes place regularly. It is the step of cleaning data. One of the goals there is identifying outliers. The methods used are mostly from machine learning, such as density clustering or regression analysis.

The solution adaptation takes place on the symbolic and the overall levels. The goal is to correct errors in the messages, as discussed below. One cannot detect such errors on the lower levels.

In speech, often one simplifies the situation by assuming that the spoken words are commands that should be executed. The queries are spoken commands and the solutions are symbolic commands that have to be performed. Even if there are not many commands, they can be formulated in many different ways that cannot all be foreseen in detail.

Here, we have two main views:

- Speech is simply a medium to contact a case base, for instance, if we have a question and want an answer.
- In speech we can also express demands or commands that should be executed automatically by some machine. One can use this in monitoring environments where persons give commands or in assisted living where persons ask for some support. For both tasks, a similarity match has to be performed.

As an example, we consider the command “Please open the window”. There are several conditions connected with this command:

- In the case base “open window” has to be available as an executable action.
- The command has to be uniquely defined. This means there is no ambiguity. This can happen if there are, for instance, two windows. In this case a feedback from the system is needed for resolving the ambiguity.
- There may be other reasons for restricting the execution of the action such as “Windows should not be opened at night”.

19.6.10 Errors

For dealing with errors one has first to observe that there are different kinds of possible errors. Often, they are difficult to identify because mostly there are no clearly defined problems with objective solutions. There is no specification, and correctness is located in the heads of humans. On the pure process level it is therefore impossible to detect errors. This is possible only if one goes to higher levels of abstraction where additional knowledge is available.

The starting point for errors is still, however, on a low level: The measurement of signals may not be recorded correctly. For discovering this, one has to go to higher levels, as shown in the example on assisted living.

However, misunderstanding does not automatically lead to an unintended action. It leads to a revised view on the error concept: Errors occur when the wrong action or decision is called.

We consider a typical example taken from an application in conversational CBR for e-commerce. For example, if a customer poses a query, there can be the different situations that have been discussed in Chap. 9, Adaptation, Sect. 9.3.3.1 on completion rules:

- (a) A query may not be physically understood.
- (b) Two queries may contradict each other.
- (c) The query may be incomplete.
- (d) The seller may want to offer a different sales object than demanded.

The performed action is a reply to the customer where one uses predefined templates. However, in classical conversational CBR it is always assumed that a written text is read correctly. But in the speech context the speech may not be understood correctly. One has to find something strange (i.e., possibly wrong) on a higher level of abstraction and has to start an intermediate dialogue for clarification.

19.7 Applications

19.7.1 *Diagnosis*

For making a diagnosis and choosing a therapy, the development of measurements over time is often crucial. The difference with static attributes is that one regards the whole time series as a unit that has to be handled by the similarity measures.

The task of using the temporal development is to discover significant sequences for depicting symbolic time series. For this, the case base is a library containing time series examples as cases. One task is selecting all qualified sequences that are non-redundant and carry useful information. This will lead to more compact information. Another task is to collect typical feature vectors of measurement sequences in the case base for giving advice on what actual measurements could mean. For this, one has to associate attributes with the time series. Usual examples occurring in diagnosis are:

- Increasing;
- Decreasing;
- Stable;
- Mean, diversity, and so on.

Often such attributes are indicators of faults.

19.7.2 *Speech Applications*

In speech recognition one distinguishes between longer speech and single-word approaches. Longer speech is used when one speaks into a microphone; the speech is then converted to written text. This does not require understanding. The commercial tools work quite well if one speaks clearly and directly in front of the microphone, i.e., if a headset is used. If this is not available, the recognition performance goes down rapidly, depending on the distance between the speaker and the microphone.

Single-word approaches (or those of a small number of words) are used in short commands.

The vocabulary container plays a special role in speech applications. Although the expressions used are to a large degree determined by the application, there is still some degree of freedom. This is because often there are different ways to formulate an expression, for instance, by using synonyms. One can take advantage of this for minimizing recognition errors. This leads, for instance, to the following advice:

- (a) Avoid expressions that are phonetically similar.
- (b) Avoid expressions that are complex and difficult to formulate.

Example applications are (some are investigated below in more detail):

- (1) Conversational CBR: Here a dialogue between the system and a user takes place. If the user is a human, this human can present queries in spoken form. The system can give the reply also in spoken form.
- (2) Monitoring machines: A human gives commands to machines that are difficult or dangerous to reach. Examples of such commands are “more heat” and “speed up”.
- (3) Controlling robots: Giving spoken commands to a robot, for instance, “turn left” or “pick up the package”.

In such applications one is not always near to a microphone and/or there is more or less noise from the environment. On the other hand, one does not have to tell long stories. It is sufficient to utter single words or phrases.

19.7.3 Conversational CBR

In Chap. 20, we discuss conversational CBR. The main point is that the query is initially not fully presented; rather, a conversation takes place to complete or modify it. For a speech approach, one needs two automated systems for the communication:

- (1) A speech recognition system that understands the speech of the user.
- (2) A speech synthesiser for talking to the user.

The vocabulary consists of terms used in the query and in the solution. From the speech recognition point of view it is important to see that whole phrases are used rather than just single words.

19.8 Tools

An extension of the system jColibri that accepts time series data is described in Bottrighi et al. (2009). For speech recognition a number of quite powerful commercial tools and tool boxes are available. The problem they have in common is noise

when no headset is used. Examples are Dragon (<http://www.nuance.com>) and IBM ViaVoice (<http://www-01.ibm.com/software/pervasive/viavoice.html>).

A toolbox for developing speech recognition systems is in Sphinx (<http://cmusphinx.sourceforge.net>). Another very comprehensive tool box is described in The HTK Book (Young et al. 2005). The Festival Speech Synthesis System (<http://www.cstr.ed.ac.uk/projects/festival>) offers a general framework for building speech synthesis systems. Festival is multilingual (currently British and American English, and Spanish) though English is the most advanced. It runs on multiple platforms offering black box text to speech. Another speech synthesis system is Mary (<http://mary.dfki.de>). It is a continuation of Festival and supports English, German and Tibetan.

19.9 Chapter Summary

This chapter is concerned with measurements of sensor data and speech in the CBR context. Both are regarded as stochastic real-valued processes with an analogue representation. Our view is that this ranges from elementary signal recognition to understanding. This falls into the local-global view used in the whole book.

This view was led by a level structure as it was also for images and text. The levels are concerned with stochastic processes, features, symbols and an overall understanding. The stochastic processes are modified in different ways until features can be extracted. In general, the features are coefficients of certain function representations or combinations of them. Features provide the representation format in which the recorded signals are stored. There are two kinds of similarity measures considered, the feature one and the symbolic one. The feature measure has access to the features. For the symbolic measure, semantics is needed because it refers to the meaning of the words.

In speech, simple methods use linear prediction as a mathematical method. Widely used features are the MFCC feature that is not easy to compute. These features are obtained for the analysis of the vocal tract. Some applications are shown that lead to a discussion of the error concept. The discussions are concerned with applications.

19.10 Background Information

The general theory and methodology of measurements of sensor data and spoken language in the CBR context is presently still somewhat restricted. This topic is investigated in very specialised applications but they do not from a principal methodological point of view. In the long term, this is likely to change. This can be seen from attempts in other contexts.

Time series have always played a role in diagnosis, medical as well as fault diagnosis. This was done, however, on a higher level of abstraction. Mostly, time

intervals have been used; see Allen (1983). Applications of the analysis of time series in medicine with CBR in the spirit of this chapter are shown in Xiong and Funk (2007); see also Funk and Xiong (2007). As mentioned in Sect. 19.8, Tools, there are commercial products for speech recognition when headsets are used. In offices this suffices for many applications. An example is monitoring of machines in a lab. This can be a more complex situation; headsets are often not tolerated and noise is present.

Besides Fourier coefficients, wavelets are widely used. For both, see Bachman et al. (2000). Speech recognition uses highly developed mathematical methods. In some sense, they try to model the human vocal tract and the human ear, which have many sophisticated mathematical functions incorporated. Also, advanced methods of signal processing are used. A problematic aspect is noise and noise filtering. An overall view is given in Rabiner and Juang (1993) and Becchetti and Ricotti (2002). As an introduction to signal processing Porat (1997), Deller et al. (2000), and Quatieri (2001) are recommended.

The central problem in linear prediction is determining the coefficients used in the prediction. There is an optimization problem where learning is involved; see Rabiner and Juang (1993). Lectures on MFCC are provided in Ganchev et al. (2005) and Logan (2000). Romanyshyn and Tkachenko (2010) compare some similarity measures for speech signal fragments, which are used for recognition; they also discuss their application and efficiency.

Maier and Moore (2009) provide an extended framework that is relevant to any similarity-based recognition systems. It applies the local-global principle in a way that puts more emphasis on the aspects of speech. The outcome is a proposal for a new approach termed Case-Based Automatic Speech Recognition. An application of CBR to speech synthesis is in Gonzalvo et al. (2007). A platform dedicated to e-learning for the visually impaired learners is shown in Azeta et al. (2009). The objective is an intelligent speech-based e-learning system with dual interface-voice user interfaces (VUIs) and Web user interfaces (WUIs). Case-based reasoning was engaged to provide intelligent services.

19.11 Exercises

Exercise 1 Describe a small vocabulary for giving commands for monitoring machines of your choice. Choose the words in such a way that they are phonetically different.

Exercise 2 Describe a monitoring situation in which measurement and speech should be combined.

Exercise 3 Define a simple language for guiding robots.

Exercise 4 Describe a medical situation where the temporal development of body temperature, blood pressure, and so on is essential for a diagnosis.

Exercise 5 Describe an economic situation where the temporal development of certain indicators is used for predictions.

References

- Allen JF (1983) Maintaining knowledge about temporal intervals. *Commun ACM* 26:832–843
- Azeta AA, Ayo CK, Atayero AA, Ikhu-Omoregbe NA (2009) A case-based reasoning approach for speech-enabled e-learning system. In: *ICAST 2009: 2nd international conference on adaptive science & technology*. IEEE, Los Alamitos, p 211
- Bachman G, Narici L, Beckenstein E (2000) *Fourier and wavelet analysis*. Springer, New York
- Becchetti C, Ricotti KP (2002) *Speech recognition: theory and C++ implementation*. Wiley, India
- Bottrighi A, Leonardi G, Montani S, Portinale L (2009) Extending the JColibri open source architecture for managing high-dimensional data and large case bases. In: *ICTAI'09: 21st international conference on tools with artificial intelligence*. IEEE, Los Alamitos, p 269
- Deller JR, Hansen JHL, Proakis JG (2000) *Discrete-time processing of speech signals*. IEEE, Los Alamitos
- Funk P, Xiong N (2007) Extracting knowledge from sensor signals for case-based reasoning with longitudinal time series data. In: Perner P (ed) *Case-based reasoning for signals and imaging*. Springer, Berlin, pp 247–284
- Ganchev T, Fakotakis N, Kokkinakis G (2005) Comparative evaluation of various MFCC implementations on the speaker verification task. In: *SPECOM 2005: 10th international conference on speech and computer*, Patras, Greece, 17–19 October 2005
- Gonzalvo X, Iriondo I, Socoró JC, Monzo C (2007) Mixing HMM-based Spanish speech synthesis with a CBR for prosody estimation. In: Chetouani M et al (eds) *NOLISP 2007: advances in nonlinear speech processing*. Lecture notes in computer science, vol 4885. Springer, Berlin, p 78
- Logan B (2000) Mel frequency coefficients for music modeling. In: *MUSIC IR 2000: international symposium on music information retrieval*, Plymouth, MA, 23–25 October 2000
- Maier V, Moore RK (2009) The case for case-based automatic speech recognition. In: *INTER-SPEECH 2009: 10th annual conference of the international speech communication association*, Brighton, United Kingdom, 6–10 September 2009
- Porat B (1997) *A course in digital signal processing*. Wiley, New York
- Quatieri TF (2001) *Discrete-time speech signal processing: principles and practice*. Prentice Hall, Upper Saddle River
- Rabiner LR, Juang BBH (1993) *Fundamentals of speech recognition*. Prentice Hall, Upper Saddle River
- Romanyshyn Y, Tkachenko V (2010) Comparison of similarity measures of speech signals fragments. In: *MEMSTECH: V1th international conference on perspective technologies and methods in MEMS design*. IEEE, Los Alamitos, p 184
- Xiong N, Funk P (2007) Concise case indexing of time series in health care by means of key sequence discovery. *Appl Intell* 28:247–260
- Young S, Evermann G, Gales M et al (2005) *The HTK book*, version 3.3, April 2005. <http://www.cs.tut.fi/courses/SGN-4507/htkbook.pdf>. Accessed 28 Feb 2013

Chapter 20

Conversational CBR

20.1 About This Chapter

CBR systems discussed so far have the problem formulation, also known as situation assessment, done in one step only. In these systems, users are given some form where they enter all the description of the new problem at once. In those systems, if the user cannot formulate a problem at once and make it entirely understood by the system, no answer can be given and no solution can be delivered. This chapter discusses CBR systems that interact with users in order to capture the problem in a conversation. The reader is expected to be familiar with Part I.

20.2 General Aspects

Problem formulation is the very first step in the CBR process model and usually a prerequisite to the remaining steps. Like other methodological processes, it does not specify how this should be done. Nevertheless, a user may not necessarily be able to formulate the problem at once in a machine-readable form that is ready for the CBR system. For this reason, a more flexible approach, Conversational CBR, employs a dialogue to obtain the needed problem formulation.

In some domains like troubleshooting and e-commerce, cases are described each with a different set of features. One direct consequence of such heterogeneity is that of capturing the new problem from the user. There need not be a predefined list of attributes that can be captured, at least one that the user knows. In fact, rather than a consequence, it is indeed the reason for the need for an alternative method such as that of using conversations.

Conversational CBR systems typically start the interaction by asking users for an initial description of the problem. The system then attempts to utilize its understanding of such a description in order to record a partial description of the query. It then uses this partial description to start a dialogue with the user. The dialogue typically consists of questions and answers. The system needs to find questions that will help determine the most useful case that will solve the user's problem in an

optimal fashion. Here, we refer to optimal as the minimum number of questions that lead to successful retrieval.

Conversational CBR systems therefore combine situation assessment with retrieval. This flexible approach, however, comes at a cost. Managing dialogues during retrieval requires new methods. Knowledge from the similarity container becomes useful for selecting the best next question to ask the user. For example, the weights of features embedded in questions can be used. Overall, the conversation, or dialogue, is the heart of the conversational process. For this reason, most of this chapter focuses on it.

The conversation process consists in an attempt to have the user answer questions whose answers will lead to filling in answer values in order to populate the new query. Depending on the level of sophistication of the conversation, the user may also ask questions and make demands.

As stated earlier, the situation where conversational CBR is applicable is when the user does not formulate the new query at once. The reasons for this vary from users not understanding the problems well to not having all the information available about the problem to simply not wanting to present it all at once.

This chapter introduces conversational CBR in more detail and then presents the four knowledge containers in conversational CBR. We will use the terms dialogue and conversation interchangeably. Additionally, we further discuss conversation systems, architectures, and evaluation. We conclude with discussions on additional issues like image dialogues and dialogue case bases.

20.3 Conversational CBR

An intelligent problem-solving tool is expected to be as flexible as human experts while formulating a problem. Human experts when asked to solve problems try not to ask one large set of questions to elicit the problem to be solved. Rather, human experts ask a few questions at a time, utilizing the answers to gradually create an interpretation of the problem to be solved. They are capable of selecting which questions to ask based on an initial exposition of the problem. This results in a conversation with the user. These conversations can be arbitrarily complex. This depends on the problem and on how much is already captured by the system. Simple forms of these interactions are standard in call centres.

Say, for example, your car will not start and you call your mechanic (or an expert in a call centre) to ask for help. It would be preposterous to have your mechanic ask you several questions without getting an initial brief description of the problem from you. Let's say your brief introduction to the problem is, "Yesterday it took me several attempts to start the car, but then eventually it started. It sputtered a bit while cold, but then worked fine and I drove home uneventfully. This morning though it did not start at all".

For humans, this process is guided by underlying processes but it is not formalized. For automating such a process, formalization becomes necessary. Such underlying processes are given, for instance, to the employees of a call centre.

An introduction to the problem contains relevant information for a diagnosis. For these situations, in order to provide computer support, a CBR system has to be equipped with techniques to capture the initial information and then interact with the user in order to acquire sufficient information to solve the problem. The goal is not to reach a human-like level of conversation performance, given that CBR is not aimed at general conversations. The goal of conversational CBR is to capture the problem by asking the minimum number of questions possible. The idea of minimizing questions relates to avoiding asking any question whose answer will not improve retrieval. This aim thus prevents users from being asked useless or repetitive questions and from perceiving their experience as dissatisfactory. It is important to keep in mind that users who find the experience dissatisfactory may quit the dialogue, preventing the system from successfully reaching its retrieval goal.

The conversational problem-solving method for interactive and incremental situation assessment consists of two steps. In the first step, the user informs a brief, initial description of the query through a static interface or form. In the second step, the system interacts with the user, dynamically proposing questions to be answered by the user.

A simple interaction between a user and a conversational CBR system is depicted in the flowchart in Fig. 20.1. As the user enters the initial description, the system fills in values, creating a partial description of the new query. The system uses the partial description to compute a partial similarity and rank cases in the case base. The system presents to the user a ranked list that includes questions, their respective cases, similarity scores and solutions. Later we will discuss how other measures and list variations can be shown. For example, the potential answers to a question can be shown to lead to their respective solutions.

Users may select a solution and end the interaction at this point. When answering a question, the system revises its partial query description and partial similarity to prepare a new list to present to the user.

20.4 Knowledge Containers

The main method where knowledge is manipulated in conversational CBR systems is that of managing conversations. Conversations are related to all knowledge containers. Consequently, we present the conversational containers without getting into specificities of dialogue management, reserving a specific section, Sect. 20.5, for the topic of conversation systems.

20.4.1 *The Vocabulary Container*

Despite the many relations between conversations and all containers, the most important container is the vocabulary container. This is because the conversation has to be conducted with the terms of the vocabulary container.

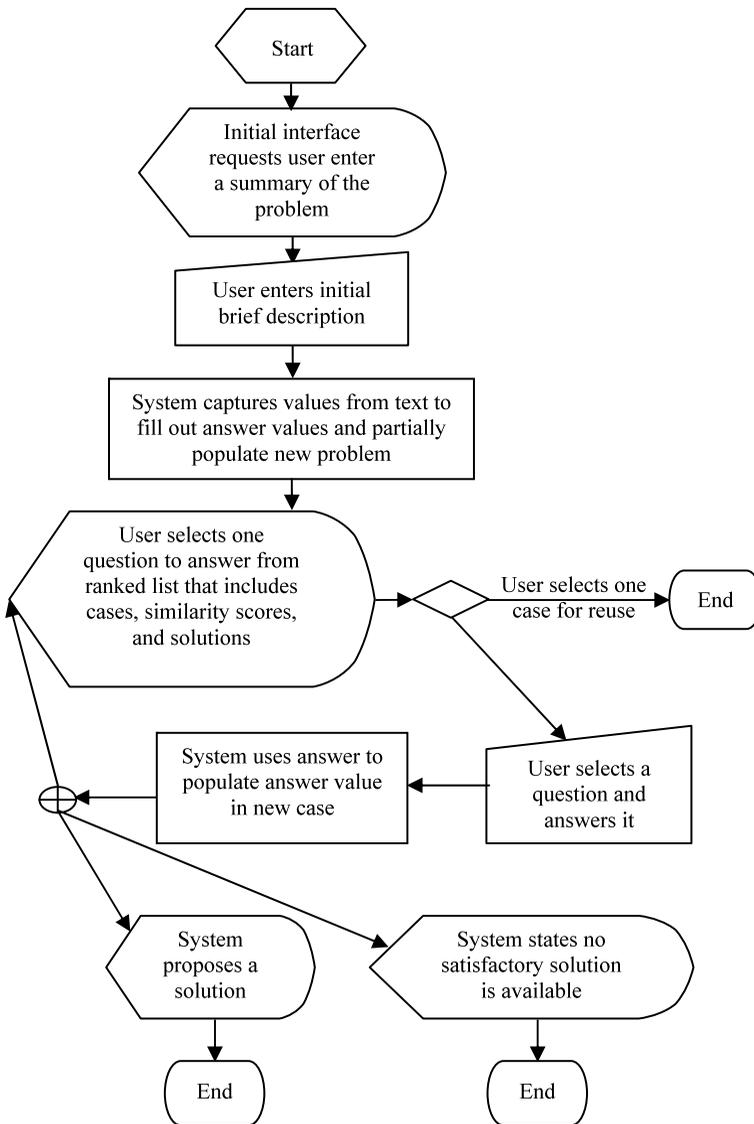


Fig. 20.1 A simple interaction

The vocabulary container of conversational systems dictates the language used in the interactions between the user and the conversational system. This refers to both the language of the initial description and the language used in the dialogues. It is necessary that the conversation output (i.e., answers given by the user to the system) be compatible with the vocabulary adopted in the CBR system.

The vocabulary container may incorporate sophisticated methods to translate or convert terms in the language of the users into the vocabulary of the cases. This may

require additional transformations but may allow users to answer questions using free text or voice. The simplest way to implement the conversational system is to make its vocabulary known to the user in the form of drop-down lists for selection. More details are discussed in the sections that follow.

20.4.2 The Case Base Container

Cases in conversational CBR are particular in that they may use question-answer pairs rather than feature-value pairs and that they may use heterogeneous indexing vocabularies. The first problematic distinction offered is the inability to use the same questions throughout the entire case base. In the examples presented in most previous chapters, an indexing vocabulary is designed for CBR systems where indices are attributes whose values can be found for every case. Heterogeneous cases are represented with different collections of indices. For example, an e-commerce application will have cases with several distinct products; the features describing articles in apparel include colour, size, and texture, whereas in electronics it may include brand, model, capacity, and so on.

20.4.3 The Similarity Container

The selection of the similarity measure for conversational CBR systems follows the same principles as those described in previous chapters. Few exceptions need to be noted.

The relative relevance of features is to be established as part of the similarity measure during system design. It is important to consider the possibility of variations of the weighting scheme after design. For example, in a troubleshooting system, a client may request to learn only about solutions that do not include fixing or replacing a given part; this could be implemented via a change in weights. In principle, this is a consequence of the application domains for conversational CBR. The fact that the user is a customer or client suggests the need for a user model. Consequently, different users may be better served with different utilities, which can be represented via different sets of weights.

20.4.4 The Adaptation Container

The situation in which a user is offered the most suitable solution from a conversational CBR system, and this requires adaptation, is not very usual in the typical domains and application tasks that typically adopt conversational CBR (see Sect. 20.9).

In e-commerce, for example, usually the cases originate from actual products and thus adaptation might lead to an inexistent product. However, it is possible that the

conversational system uses a reduced set of cases and adaptations are needed. One way to proceed when a user requests a modification to a case is to search locally for alternative values for specific features.

Adaptation may also be desired when the conversation leads to user frustration. It may happen that a dialogue termination strategy recognises that no other answer would retrieve a valid case for a user. Sometimes this is called query failure. These situations pose the need for adaptation of the entire dialogue, such as changing the answer to a previously answered question. One of the directions to pursue is then to create a new dialogue cycle by demonstrating to the user how many cases become available after one answer value is changed.

20.5 Basic Conversation Systems

In this section, we detail methods that can be used to manage conversation systems.

20.5.1 Processing the Initial Description

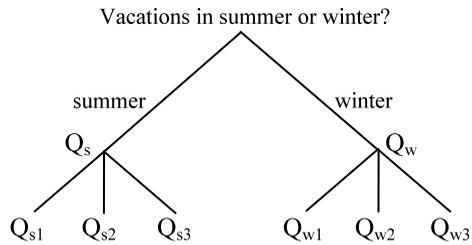
It is typical of conversational CBR systems to provide an initial step prior to starting the dialogue. This step consists of asking the users to submit a short and likely incomplete description about their problems. The system uses this description to attempt assigning values to case features. The partially populated case thus becomes accessible to dialogue management.

The choice of methods depends on the nature of the feature. If all features have single values, then a simple comparison between words in the description and all allowable values of case features suffices as a first step. It will not suffice if case values are expressions that can vary in form.

Depending on the complexity of the entered description, this step will have its proportional difficulty. The simplest approach is to take the description as a bag of words (see Chap. 17, Textual CBR, Sect. 17.3.3) and match the words with values of features in case descriptions. Nevertheless, as bag of words is a textual CBR method, any other textual CBR method can be used in this step. Intermediary levels of difficulty include expansion (see Chap. 17, Sect. 17.3.7) with synonyms, such as word sense disambiguation. At the extreme of complexity are deep natural language processing and the use of distributed representations (see Chap. 17, Sect. 17.3.5).

20.5.2 Dialogue Management

Dialogues for conversational CBR can vary in the level of sophistication. In the simplest form, only the system asks questions and the user can only select an answer

Fig. 20.2 Dynamic form

from a drop-down list. By augmenting reasoning systems with dialogue functionalities, conversational interfaces can facilitate collaborative, mixed-initiative interactions with problem-solving abilities. In the broadest formulation of the problem, conversational agents engage in spoken dialogues. This includes speech recognition and speech synthesis and is discussed in Chap. 19, Sensor Data and Speech. This section introduces elements that are general to many levels. For example, we discuss systems where both the system and the user can ask questions.

The dialogue is a sequence of actions that need organisation and management. For this reason, we will discuss a number of principal questions connected with dialogues and conversations.

A dialogue can be seen as a template—a structured form that includes a set of features where one or both partners know the features and how they can be valued. The features in these forms can be static and ordered, or they can be dynamic. More on templates and dialogue architectures is given in Sect. 20.6.

In computational systems, one partner is the user and the other partner is the system. At each iteration of a dialogue, requirements on both sides may change. It is important to keep track of requirements on both sides to successfully manage a dialogue.

Dialogues are dynamic and, in principle, they are an extension of dynamic forms. An example of such a form is in Fig. 20.2, where the answer to a question directs the focus of the dialogue to a subset of questions in a tree. If the answer to the question is “winter”, then the dialogue is directed to the questions on the right that follow the path from *winter*, and vice versa.

Dialogues take place between partners. Roles of the partners describe who is active or passive. Each partner in the dialogue can play an active or a passive role and these roles can change. Both partners can raise questions, make demands and give answers. In the example in Table 20.1 the active part is the user. The user enters questions and demands. The system is passive because it is simply reacting.

The actions performed by partners in a dialogue are as follows:

- (a) Active partners can present:
 - Questions,
 - Demands,
 - Information without being asked (*proactive*).
- (b) Passive partners can present:
 - Answers to questions and to demands.

Table 20.1 A simple dialogue

Possible questions from users	Possible answers from system
?-domain (attribute)	domain (attribute)
?-specific attribute-values available	yes, no, partial list, approximate values, constraints
?-solution properties	relevant properties
?-solution has property	yes, no, degrees, constraints
?-solution has functionality	yes, no, degrees, alternatives
?-which solution has functionality	product, list, many, none, possible, degrees
?-combination of properties	yes, no, constraint, products

Table 20.1 shows an example of a dialogue, which we analyse next.

- (a) This is a typical dialogue occurring in e-commerce. It is simple insofar as the user has no choice of the vocabulary. Every term is prescribed in a multiple-choice situation.
- (b) The user raises questions that are answered by the system with useful answers.
- (c) Each comment of the system is understandable to the user.
- (d) The answers to the questions are compatible with the case vocabulary. This cannot be seen from the outside. The system designer has to ensure this is done.
- (e) The order of the questions has to be decided. The goal is to make the dialogue as short as possible and to avoid unnecessary questions. We will discuss the ordering of questions later.
- (f) This looks like a fixed list of queries; actually, it shows only questions and answers that occurred in an actual dialogue. What the table does not show is that there is a certain degree of freedom. This includes the fact that there may be additional user questions depending on the answers of the system.

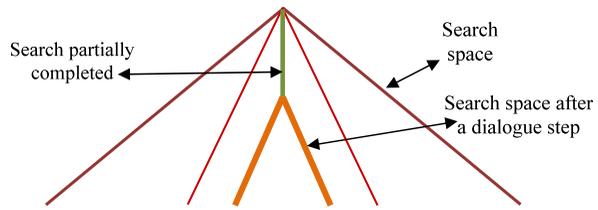
20.5.3 Dialogue Formalisms

For machine support and integration into a CBR system, several concepts have to be formalized. The general concepts and operations to be formalized are:

A situation is a vector **sit** with the following components:

- (1) Roles with the values {active, passive}.
- (2) Information situations: Describing what we know about the user. This has two sources. One is a priori knowledge and the other is knowledge obtained from the conversation. This knowledge tells something not only about the problem in question but also about the user. The latter can influence subsequent system actions.
- (3) Dialogue actions {queries, demands, replies, interrupts, finish}.
- (4) Results of actions: New information situations.

Fig. 20.3 Shrinking the search space



- (5) Selections of roles, dialogue actions. A special action is finish. When done by the user, it usually comes without comments. If done by the system it indicates the system has understood the complete query. The dialogue is continued only if necessary.
- (6) Strategies as mappings:

$$\{\text{Set of situations}\} \rightarrow \{\text{Set of actions}\}.$$

We consider two types of conversations with respect to the case base:

- (a) Conversations for getting support to solve a single complicated problem. Here one does not deal with a large case base.
- (b) Conversations for navigating through a large case base. This happens, for example, in e-commerce applications.

If in a sales or troubleshooting dialogue the case is described by attributes and their values then it is natural to present these attributes to the customer and ask for desired values. Although in principle the dialogue is quite simple, there are still a number of conditions for a successful dialogue. The main ones are that expressions are short and understandable to the user. The risk is that the users become dissatisfied and quit before the goal of the conversation is achieved.

Hence, the dialogue has to focus on two aspects simultaneously: To obtain more insight into the context of the user and to come closer to the goal of the conversation. This is illustrated in Fig. 20.3 and Table 20.1.

While the questions themselves are determined by the attributes, the order in which the questions are posed can be chosen, and this is crucial for the success of the dialogue. Unlike as in decision trees where there is a fixed order of questions, in a dialogue there are no prescribed ordering. One can flexibly choose the questions.

20.5.3.1 User Information

There are restrictions and conditions a successful dialogue has to respect. The first type of restriction is with respect to the user context. This requires a formal *user model*. For a conversation, we define:

Definition 20.1 A user model is characterised by:

- (1) Which expressions does the user understand?

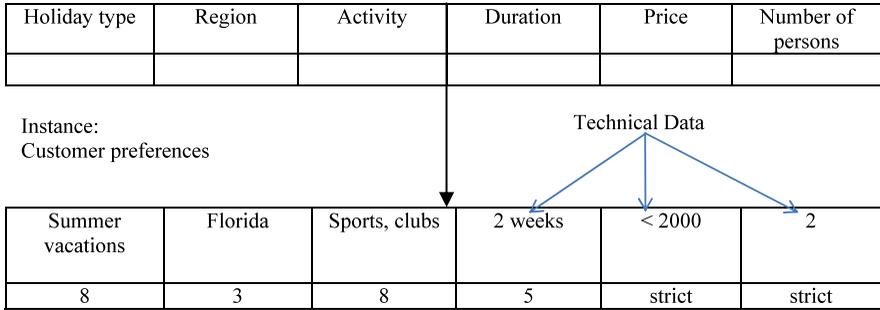


Fig. 20.4 Travel agency

- (2) What level of detail or abstraction is helpful and accepted?
- (3) Which queries make sense to the user?
- (4) What has one to know about user preferences?
- (5) What conversation style should be used?

One of the components of the situation vector is information about the user. As an example of obtaining information about the user, we consider a travel agency. Figure 20.4 contains a top row with alternatives (e.g., region, duration) through which the customer can scroll to select a value. The choices made by the user are in the bottom row table. In addition, the customer can add the importance in the form of a weight.

This multiple-choice selection restricts the products in which the customer can be interested. Can one also draw further conclusions about the customer that influences the search space? Consider the situation obtained so far:

- There are six attributes with symbolic or numeric domains. On each domain a local similarity function has to be defined. Strict requests have a filtering function.
- There are no problems to understanding the attributes. Additional attributes for the supplier, such as travel code, are of no interest to the customer.
- The customer will also have no difficulty in expressing preferences using weights.
- From the selection and the weight given for sports and clubs, we can associate the customer with potential demographics. The answer to “number of persons” suggests that individuals travelling can be a couple.

These observations can help enhance the information about the user. The additional information may be helpful for achieving additional goals. In the situation of an e-commerce application for a travel agency, it is likely that the agency also has other products to offer, such as tickets to a pop concert. In this situation, the additional offers should match the target demographics of the products. Along the same lines, products targeted to a different age group would not be offered.

Methods for expansion are the ones used in information retrieval and in Chap. 17, Textual CBR. It is important to observe here that such expansions should be limited to the language that is allowable in case feature values. This is easy when the domain is well bounded and technical.

20.5.4 Lengths of Conversations

Conversational CBR systems have two main goals. The first is shared with all other case-based systems, to retrieve the solution that provides better utility to the user. The second is unique to conversational CBR; it is to make the dialogue as short as possible.

A general guideline for defining the lengths of conversations is to stop when no further information can be obtained. This also applies for getting problem formulations in systems where we present a sequence of attributes and ask the user to fill in the attribute-values (as discussed in previous chapters). A somewhat refined way (done by several CBR systems) is to stop the questions if the remaining values are not needed. This does not, however, guarantee the shortest dialogue. Everything boils down to the problem, “Which question should I ask first or next”?

To illustrate this we consider asking questions to capture attribute-values as a race of the cases. Suppose only one attribute-value is known. Depending on the weight and the local similarity of the known attribute, cases can be sorted based on the partial similarity score. The case with the highest partial similarity score is ahead in the race.

By adopting this approach, each iteration or cycle will produce a new ranking of the cases. If there is a step where no other cases can reach the leader independently of the results of the remaining questions, then this leader is the winner.

At this point we have to distinguish two different utilities:

- (1) The goal is to have as leader the case with the highest similarity (i.e., the nearest neighbour).
- (2) The goal is to have as leader a case with a sufficiently high similarity.

There is no unique way to reach these goals. The length of the dialogue, and consequently the selection of the next question, is influenced by two aspects:

- The value that is the expected information to be answered;
- The importance (i.e., weight) of the attribute addressed in the query.

20.5.4.1 Question Selection

Multiple methods can be used. A very simple one would be to simply select the question that asks for the attribute with highest weight. A more elaborate method consists of choosing the attribute with the highest expected information content, as is done when constructing a decision tree.

The crucial notion is the entropy (see Chap. 22, Basic Formal Definitions and Methods) of an attribute A with $\text{dom}(A) = \{a_1, \dots, a_n\}$:

$$\text{Entr}(A) = - \sum_{i=1}^n P(a_i) \cdot \log(\text{Prob}(a_i)).$$

The first remark is that the situation is not the same as in decision trees. There, one has a classification problem, but here we have cases or solutions where the user may have a higher or lower utility. Therefore, it does not seem justified to exclude cases that miss just one attribute-value.

In addition, asking for the value of the attribute with highest entropy neglects the importance of the attributes. This may result in asking for values of features with low weights, which might cause cases with low similarity value (i.e., low utility) to lead the race. Another problem is that some aspects of the utility of the user may be already known.

One approach suggests that one way to select the next question is to exclude all cases whose values do not match the attribute-value under consideration. This view leads to the extended expected information gain *extgain* of A , where A is the selected attribute. CB is the case base, c is a case and $A(c)$ is the value of a case c :

$$\text{extgain}(s, A) := - \sum_{a \in \text{dom}(A)} \text{Prob}(a) \cdot \log_2 \left(\frac{\sum_{c \in C} \text{sim}(A(c), a)}{|CB|} \right).$$

Here, all cases are considered but are not counted equally. A further refinement is obtained by using the variance of similarities. Higher similarity variance in the set of candidate cases implies better discrimination between possible target cases and unsatisfactory ones. The *simVar* measure is defined as:

$$\text{simVar}(q, A, a, CB) := \frac{1}{|CB|} \cdot \sum_{c \in C} (\text{sim}(q_{A \leftarrow a}, c) - \mu)^2.$$

Here the value of an attribute A is a , q is a query and $q_{A \leftarrow a}$ denotes the result of assigning the value a to the attribute in the query q ; μ is the average value of all similarities, and CB is again the case base.

simVar is based on the variance of similarities. This problem is related to diversity, as discussed in Chap. 14, Advanced Retrieval. One can say that none of these methods is optimal in all situations, but that the simple information content approach is outperformed.

20.6 Architectures for Dialogues

In this section we discuss architectures for dialogues. The goal is to enable systems to perform conversations in a systematic way. Thus, we need domain-independent architectures that can provide the functionalities required. Here, the term speech act is used in a very general form.

There are three major types of architectures for performing dialogues:

1. Graph-based architectures, where the graphs are directed. They have the advantage of being well structured because one has to follow the direction of the arrows; diversifications are allowed.

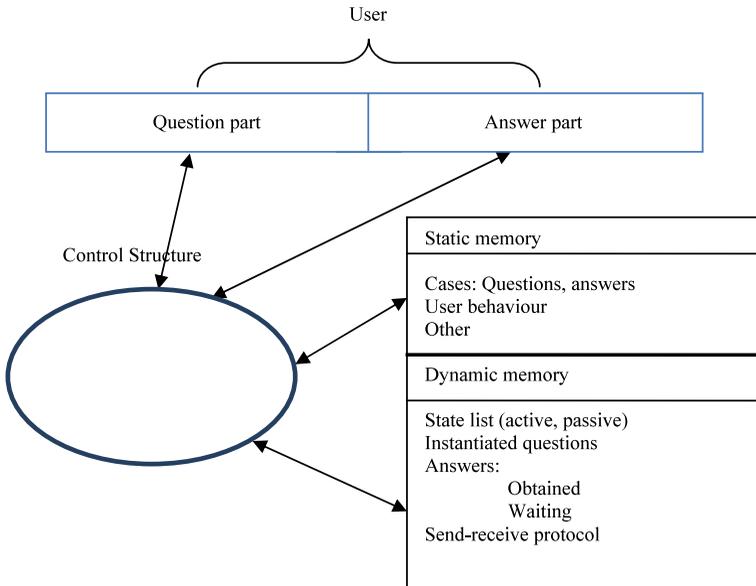


Fig. 20.5 A dialogue architecture

2. Template filling architectures employ a feature vector with values to be determined during the course of the conversation. They allow a broader range of conversations and do not impose the ordering restrictions that graph-based systems do. The systems discussed above that raise questions for getting attribute-values fall in this category.
3. Plan-based architectures are more general than the other two approaches. In these, the conversation follows a general plan. In such systems, there is a goal and some allowed steps to proceed. Planning applications are discussed in Chap. 9, Adaptation.

In Fig. 20.5, we describe a generic architecture of the template filling type in some detail. The user interface as presented to the user is a window that has two parts that are opened and closed dynamically:

1. The user part
2. The system part

The architecture can be refined in various ways. In the static and dynamic memories one can store knowledge needed for the dialogue. If natural language is used, this can, for instance, be a thesaurus; see Sect. 20.8.1.

Next, we will present a more general approach. This is the Discourse Goal Stack Model (DGSM), which is CBR-oriented. It presents a form of goal-oriented dialogue where the central task is selecting useful cases. DGSM builds on the observation that there is symmetry between the discourse goals that trigger CBR and the discourse goals that arise in CBR when the facts of a problem description are being

elicited. When a system is engaged in a dialogue with a user, the user may make direct or indirect requests for information that can only be answered if the system elicits additional information from the user. This occurs, for example, if the user requests information for diagnostic or product selection. DGSM can be viewed as a kind of graph-based architecture.

DGSM consists of:

- A goal stack;
- A collection of discourse goal types;
- A forest of augmented transition networks (ATNs) in which nodes are discourse goals and arcs are speech acts by the user or the system;
- A goal handler, which is responsible for determining the appropriate action to take in response to the goal at the top of the stack. The goal handler selects from among the following actions, based on the value of the current top of the stack and the most recent speech act by the user.

The DGSM's main steps are:

1. If the current goal corresponds to a node in an ATN and the user's utterance is recognised because it matches a transition from that node, the goal handler pops the stack and pushes the node at the end of the transition.
2. If the current goal corresponds to a node in an ATN but the transition contains a speech act by the system, the speech act is generated, the stack is popped, and the top of the stack is replaced by the node at the end of the transition.
3. If the user's utterance does not correspond to a transition from the state at the top of the stack but matches an initial transition in another ATN, the utterance is interpreted as a change of topic. The state at the end of the transition is pushed onto the stack.
4. If the state at the top of the stack is the end state of an ATN, then it is popped.

20.7 Quality and Evaluations of Dialogues

Quality can be associated with each conversation in various ways. It is difficult to measure because it is to a large degree context-dependent. Nevertheless, every conversation or dialogue has certain goals. The quality determines how far these goals are reached. This is what the evaluation is supposed to measure. Here we distinguish two kinds of evaluations, shallow and deep, which we describe next.

Shallow evaluations: These are concerned with measurable criteria that are directly of interest to the system designer. These criteria determine the success of conversation systems. However, they cannot be directly influenced. Shallow evaluations are concerned with user acceptance. They can be measured in three ways:

1. How often do users quit the conversation before it comes to an end?
2. How often do users complain about the conversation?
3. How often do the users miss a good solution?

Table 20.2 User dissatisfaction

Reaction	Info gained	User dissatisfied	Utility (U(A,s))
Easily answered	Yes	No	Info(A,s)
Answered with problems	Yes	A little	$\frac{1}{2}$ Info(A,s)
Did not apply	No	Maybe	0
Not understood	No	Yes	-Info(A,s)

Deep evaluations: They investigate criteria that are responsible for success or failure of the shallow criteria. They can be influenced directly by the systems user.

They have to describe the underlying reasons for unsatisfactory shallow evaluations. Intuitively, there are two main reasons:

1. The contributions of the system are not accepted:
 - a. The terms used are not understood.
 - b. The reasons for the system messages are not understood.
2. The conversation is too long.

These reasons can be to some degree deduced from the user reactions. We consider two possible user reactions.

- (1) The obtained information is accepted. User is satisfied.
- (2) User is not satisfied. This may be due to:

- Terms not understood.
- Purpose not understood. Complaints about the usefulness of the message.

We use a simple numerical method in Table 20.2 to quantify the user acceptance.

Overall, an important aspect is that the system's answer to a question asked by the user should be both true and adequate. That is, not simply a true statement; this is essential for user acceptance.

20.8 More on Dialogues

20.8.1 *The Role of Thesauri and Ontologies*

The use of thesauri or ontologies is to support converting language into the adopted vocabulary. This is required when users have choices of words rather than being limited to one option, as in a drop-down list. The use of ontologies in this chapter is similar to what has been discussed in other chapters (e.g., Chaps. 12 and 17).

One difference is in the use of ontologies for conversations. In conversational CBR, their role is to make a conversation more efficient and reduce the knowledge engineering effort for building such systems. This is supposed to overcome problems when dialogues are constructed manually.

Fig. 20.6 A question in the form of a picture



Specific types of ontologies can be used, such as task ontologies and query ontologies. When using query ontologies, the idea is to organise queries in a taxonomy going from general queries to more specific ones. Then each path in the taxonomy corresponds to the sequence of queries in some dialog. This supports top-down dialogues going from general to more and more specific issues. See Chap. 12, Advanced CBR Elements.

For an example, consider a cell phone sales conversation. The system does not need to include specific information of all cell phone models available, particularly because they change often. An ontology for cell phone concepts (e.g., smartphone, camera, memory, app) and models of phones guides the system to make proper offers and determine subsequent questions to the user. The cell phone example is presented in the next section.

20.8.2 *Images in Dialogues*

A dialogue system may not only use text or formulas. It can also present images. These are discussed in Chap. 18, Images.

The next example is of a very different type. It shows how a user can be guided through the selection of a cell phone. The user may start with this question: “Which offer is the most suitable to me”? This is a rather incomplete demand!

The first query of the system is in Fig. 20.6: “Which of the following images best describes how you will be using your phone”?

The question is shown as a picture! The user understands it without a formal definition. The presentation of the picture covers a complex context that is difficult to describe verbally. The user grasps the intention easily.

Question:	Will you use the phone mainly during business times?
Answer:	Yes
Question:	How many minutes do you expect to spend on calls per day?
Answer:	120
Question:	What is your monthly cellphone allowance (in Euro)?
Answer:	500
Question:	Do you need to control the cost of your usage?
Answer:	Yes
<p>Solution: You are a business user with a cellphone allowance of €500 per month. We are pleased to recommend Talk 100+, which we feel will meet your needs. To see more click here.</p> <p>Here the offer of special numbers with low costs. </p>	

Fig. 20.7 Cell phone sales conversation

Figure 20.7 depicts the remaining dialogue using words. The last questions complete the information for selecting the product. They are meaningful to the customer because one knows that such information is important. No question is redundant. The last question is used for the form of payment and is also considered as useful.

20.8.3 Dialogue Case Bases

When a dialogue has to be performed it can also make use of experiences with dialogues in the past. For this, the case base has to contain cases with information relevant in this respect.

The case in such a case base contains the following elements:

- the query,
- the selected questions,
- the answers,
- the solution description,
- the outcomes and termination of the dialog.

Dialogue case bases are useful if the users are from a highly specialised area where sophisticated terms are used.

20.9 Domains, Applications, Commercial Use

In conversational CBR, commercial success has motivated research. The method has been particularly useful for troubleshooting where users interact with the system by answering questions that will help the system select the most useful case to retrieve.

Sales conversations are frequently the topic of conversational CBR. Traditionally, they take place between humans in stores, who are experienced to talk to customers. In an electronic conversation, this is quite different because one does not see the partner; thus electronic conversation in conventional everyday language can be fairly difficult.

We illustrate this by comparing human and machine dialogues on a principle level. As an example, we consider sales dialogues.

Conventional sales dialogue: the supplier is a human (i.e., sales assistant).

- The supplier tries to find out about the customer's requirements and provides information about appropriate products.
- The dialogue can contain all elements of communication (e.g., audio, visual) and interaction is very important.
- The supplier's knowledge from preparatory training, personal experiences, and logical conclusions may not be completely informed and/or he or she may commit errors.
- The dialogue can easily be adapted to different customers.
- The supplier is only available during certain times of the day.

Systems sales dialogue: The supplier is represented by a software agent.

The sales knowledge is in:

- Combining product-describing and functionality-describing attributes.
- Logical relationships between components, the resulting consequences, and the processing strategies for the sales dialogue are part of supplier's knowledge.

The customer will only continue dialogue with the software agent as long as the relationship between the answers and the inquiries can be noticed and the dialogue is not too long. One weakness is that adaptation to individual customers is quite complex.

20.10 Tools

Almost all CBR systems allow at least a rudimentary form of conversation, mostly by using templates. We have not found any currently available tools, but there are several running systems in the industry.

20.11 Chapter Summary

This chapter deals with situations where the user does not present a complete problem in the first interaction with a CBR system. In conversational CBR systems knowledge is manipulated via the management of conversations. Conversations are related to all knowledge containers, but the most important one is the vocabulary

container. This is because the conversation has to be conducted with the terms of the vocabulary container.

Dialogues for conversational CBR can vary in the level of sophistication. The dialogue is a sequence of actions that needs organisation and management.

For machine support and integration into a CBR system, we mention some of the concepts that need to be formalized.

Conversational CBR systems have two main goals. The first is shared with all other case-based systems, to retrieve the solution that provides better utility to the user. The second is unique to conversational CBR; it is to make the dialogue as short as possible.

Multiple methods can be used for question selection. A general guideline for defining the length of conversations is to stop when no further information can be obtained.

There are three major types of architectures: Graph-based architectures, template filling architectures, and plan-based architectures. For building a conversational system we present an elementary generic structure and a more sophisticated one.

The quality of a conversation is mainly measured by the time to when the user is annoyed and quits the conversation without coming to a decision. We provide evaluation measures for it. We include metrics in shallow and deep evaluation methods for dialogues.

The chapter includes sections on thesauri and ontologies, images in dialogues, dialogue case bases; and a discussion of domains, applications, and commercial use of conversational CBR.

20.12 Background Information

A recent overview of conversations in CBR is given in Aha et al. (2005). Though most work in this field is recent, the logic of questions and answers has a long tradition. A basic reference is Belnap and Steel (1976).

Publications discussing conversational CBR became more frequent after 1997. See, for example, Aha and Breslow (1997). Yang et al. (1997) describe a conversational system that allows the user to adapt a proposed solution through a constraint satisfaction process.

Aha and Breslow (1997) introduce their NaCoDAE (Navy Conversational Decision Aids Environment) tool for conversational CBR. In 1998, Aha, et al. showed the use of natural language processing for applying the user's initial description to assign values to the new query. Shimazu (1999) describes how to go from human conversations to automated ones.

Aha et al. (2001) investigate evaluations. Special aspects about recommender system are discussed in Reilly et al. (2005). The length of dialogues is described in detail in Schmitt (2002).

Cycopath has created several dialogue systems. The cell phone example was taken from one of their projects. Unfortunately, they no longer have a presence on the Internet.

The Discourse Goal Stack Model (DGSM) is a strategy for dialogue management described in Branting et al. (2004).

In Gu and Aamodt (2005) an evaluation technique for CBR systems, including those with conversation, is provided. An essential aspect is that it uses many simple and a few complex systems. The role of ontologies is described in (Gómez-Gauchía et al. 2006).

20.13 Exercises

Exercise 1 Select a small case base (e.g., six to eight cases) with no more than six features per case. Randomly select three cases and manually perform three iterations of selection of questions and answers. Remove the selected case from the case base. Randomly select one feature whose value is given. Use entropy and simVar to select the next question. Continue until the most similar case is found. Repeat this for the three cases.

Exercise 2 Suppose you have a music shop. Select a customer class and develop a template with questions and answers for online customers.

Exercise 3 Develop two dialogue templates for selling PC equipment, one for a computer expert and one for a layperson.

Exercise 4 Take a medical situation where several investigations have to be performed. These may be costly and painful. Describe such a situation and give an ordering on the investigation that minimizes costs and pains.

Exercise 5 Describe a situation in a call centre where the operator has to talk to the customers with problems.

- (a) Formulate guidelines for the dialogue in ordinary language.
- (b) Formalize the guidelines for storing them in an experience factory.

References

- Aha DW, Breslow LA (1997) Refining conversational case libraries. In: Leake DB, Plaza E (eds) ICCBR 1997: case-based reasoning research and development. Second international conference on case-based reasoning, Providence, RI, July 1997. Lecture notes in computer science (lecture notes in artificial intelligence), vol 1266. Springer, Berlin, p 267
- Aha DW, Maney T, Breslow LA (1998) Supporting dialogue inferencing in conversational case-based reasoning. In: Smyth B, Cunningham P (eds) EWCBR-98: advances in case-based reasoning. 4th European workshop on case-based reasoning, Dublin, Ireland, September 1998. Lecture notes in computer science (lecture notes in artificial intelligence), vol 1488. Springer, Berlin, p 262

- Aha DW, Breslow LA, Muñoz-Avila H (2001) Conversational case-based reasoning. *Appl Intell* 14(1):9–32
- Aha DW, McSherry D, Yang Q (2005) Advances in conversational case-based reasoning. *Knowl Eng Rev* 20(3):247–254
- Belnap ND, Steel TB (1976) *The logic of questions and answers*. Yale University Press, New Haven
- Branting LK, Lester JC, Mott B (2004) Dialogue management for conversational case-based reasoning. In: Funk P, González-Calero PA (eds) *ECCBR 2004: advances in case-based reasoning*. 7th European conference, Madrid, Spain, August/September 2004. Lecture notes in computer science (lecture notes in artificial intelligence), vol 3155. Springer, Berlin, p 77
- Gómez-Gauchía H, Díaz-Agudo B, González-Calero PA (2006) Ontology-driven development of conversational CBR systems. In: Roth-Berghofer TR, Göker M, Güvenir HA (eds) *ECCBR 2006: advances in case-based reasoning*. 8th international conference, Fethiye, Turkey, September 2006. Lecture notes in computer science (lecture notes in artificial intelligence), vol 4106. Springer, Berlin, p 309
- Gu M, Aamodt A (2005) Knowledge-intensive method for conversational CBR. In: Muñoz-Avila H, Ricci F (eds) *ICCB 2005: case-based reasoning research and development*. 6th international conference on case-based reasoning, Chicago, IL, USA, August 2005. Lecture notes in artificial intelligence, vol 3620. Springer, Berlin, p 296
- Reilly J, McCarthy K, McGinty L, Smyth B (2005) Incremental critiquing. *Knowl-Based Syst* 18(4):143–151
- Schmitt S (2002) *simVar: a similarity-influenced question selection criterion for e-sales dialogs*. *Artif Intell Rev* 18:195–221
- Shimazu H (1999) Translation of tacit knowledge into explicit knowledge: analyses of recorded conversations between customers and human agents. In: Aha DW, Becerra-Fernandez I, Maurer F, Muñoz-Avila H (eds) *Exploring synergies of knowledge management and case-based reasoning: papers from the AAAI workshop*. Technical report WS-99-10. AAAI Press, Menlo Park, p 81
- Yang Q, Kim E, Racine K (1997) *CaseAdvisor: supporting interactive problem solving and case base maintenance*. In: *Practical use of case-based reasoning workshop, IJCAI 1997 workshop program*, Nagoya, Japan, 25 August 1997

Chapter 21

Knowledge Management

21.1 About This Chapter

This chapter aims at educating readers on the potential benefits that CBR can offer to help identify, evaluate, capture, store, and retrieve an organisation's knowledge assets. Understanding of all previous chapters is desirable, but not crucial.

21.2 General Aspects

In a wider sense, KM is about managing certain kinds of knowledge. So, why would a book about case-based reasoning include a chapter on knowledge management (KM)? An answer to this question is the main purpose of this chapter. First, we examine KM problems, the nature of its processes, its goals and its cycle. Then we compare the CBR and the KM cycles. Next, we illustrate how methods from the CBR methodology can be used to implement KM processes. Last, we discuss for what KM tasks CBR should be used.

Knowledge management (KM) concerns methods that aim at organising, coordinating, planning, commanding, and controlling knowledge assets in an organisation. Because organisations can vary from a small team to hundreds of thousands of members, KM appears in multiple scales. Knowledge also has its several facets, making KM an even broader field. Some specific areas such as Library and Information Science (LIS), whose professionals are in charge of managing knowledge in libraries, have a strong and large agenda for KM. Another field with a wide role is Management and Organisational Science, as it comprises specialists in organisations. Other fields playing a significant role in KM are the computing fields such as Computer Science, Information Systems, Software Engineering, and Information Technology, with the role of implementing computational solutions for KM. This last facet is what we present here, specifically, how to use CBR to perform multiple KM processes.

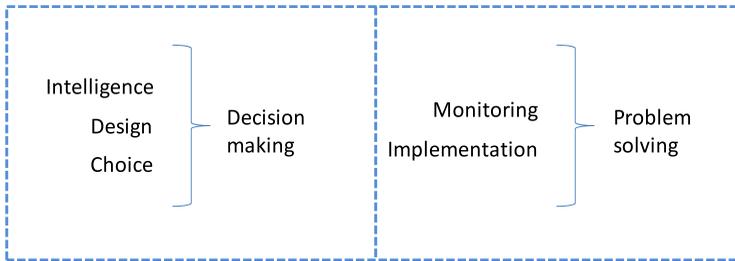


Fig. 21.1 Decision making and problem solving simplified model

21.3 Knowledge Management

We now want to introduce KM in a bit more detail. Because of the dependency on the concept of knowledge in KM, we will discuss the term *knowledge*. Next, we will associate knowledge with decision making. This is a way to model the context of knowledge, decisions, and problems. Next, we present a few KM problem situations, the nature of KM, its processes and its cycle.

21.3.1 Knowledge and Knowledge Management

Knowledge management is mostly considered as a part of general management in organisations. From this point of view, knowledge is considered as an abstract collection of assets. As with other managed assets, it has to be made clear how the knowledge is obtained, formulated, stored and used for different purposes.

Knowledge has no unique and precise definition; it is used in different perspectives. The literature suggests six different perspectives to conceptualize knowledge. In this chapter, we are mostly interested in and will be using the perspective of, knowledge as that which enables the use of information to make a decision. In knowledge management, decisions are made to deliver organisational processes.

21.3.2 Knowledge and Decision Making

Consider the model of decision making and problem solving given in Fig. 21.1. Decision making comprises three steps. *Intelligence* refers to gathering information about the problem. *Design* is about identifying what approaches could be used to reach a decision that will enable the originating problem to be solved. It is in the *Choice* step that one approach is chosen, which entails determining the potential outcome of each approach so that the one with best expected result is selected. Both *implementation* of the approach and *monitoring* are part of problem solving.

The steps in this model of decision making use knowledge in different ways. Knowledge may be used to gather information during intelligence. It is used to exe-

cute the design step because it takes knowledge to recognise when an approach has the potential to solve a problem. The choice step in the model also uses knowledge. It entails the prediction of results with the comparison between potential benefits and potential disadvantages.

The problem-solving model allows us to recognise the use of knowledge in knowledge management problems, making it easier to understand the use and reuse of knowledge in the CBR methodology. Next, we discuss some KM problems where KM steps can be implemented to solve knowledge-related problems.

21.3.3 Some Knowledge Management Problems

Consulting companies typically employ qualified personnel to provide knowledge services to their clients. A typical problem is that these companies do not document what they know.

Manufacturing organisations have problems with organizing knowledge about machinery. Experience determines when to stop machinery for maintenance. Members who have this knowledge need to share what they learn so this individual knowledge becomes organisational.

Not specific to any organisation is the need to search for answers to questions or for solutions to problems. Here we refer to the search for answers on topics that are unknown to the searcher. Searches that today are conducted in Web-based search engines not long ago were conducted exclusively by reference librarians. Effectively searching the Web or digital libraries, despite seeming mundane today, requires knowledge of the field and of the resource. Common areas in which laypersons need help are medicine and law.

An important audience of KM solutions is that of scientists. Their work entails production of knowledge and therefore they can immensely benefit from knowledge sharing. Consider identifying open research problems in a field or building a complete literature review on a topic. Given the current view of interdisciplinary scientific challenges and of how collaboration among scientists is viewed as a requirement, sharing of scientific knowledge is a major problem. The other task involving scientists thus becomes knowledge leveraging.

This problem can be tackled with KM. The main steps required are capture, store, and represent knowledge so it becomes available for distribution and reuse. These steps then enable KM processes, commonly referred to as KM tasks, knowledge sharing, leveraging, and organisation.

21.3.4 Knowledge Management: An Organisational Discipline

Knowledge management (KM) inherits the vagueness of the concept of knowledge because it targets the management of knowledge assets. KM is an organisational

discipline because it is only needed when more than one individual is involved. Individuals are equipped with internal KM processes, which are apparently seamless. For example, humans do not need an external process to share knowledge with themselves. Most humans are able to remember that touching a very hot surface will burn their skin. This is a form of sharing with oneself, in analogy to sharing between two persons.

Earlier we stated that problems solved by the use of knowledge in KM are organisational in nature. The implication to the model shown in Figure 21.1 is that originating problems are organisational and each step of decision making and problem solving requires some form of knowledge. This is an extremely important observation.

Consider that there is an entire field of study dedicated to decision making. Now consider that every organisation—be it for profit or non-profit, private or governmental, even as informal as a group of friends planning a trip—will reach its goals by executing processes. Many of those processes require decision making. The amount of knowledge involved in reaching goals is thus larger than one can imagine. Now consider that all this knowledge *should* be stored, represented, and ready to be distributed and reused. At least it *should* to the extent that the organisations do not want to make wrong decisions or to reinvent the wheel. Granted that for the friends planning a trip this does not mean much; but for professional organisations, making the wrong decisions or reinventing the wheel means waste.

The overall aggregated ability of an organisation to make decisions and solve problems is reflected in its experience curve. The more experienced an organisation's members are, the better their decisions are likely to be when delivering organisational processes. Recording experiences and providing them when applicable is what is done with CBR.

Through implementing KM steps such as creating, distributing, and reusing knowledge, the target organisation will be more likely to reach its goals and thus achieve its mission. This is because KM steps support other managerial units in better executing their own processes. Consequently, it is of utmost importance that the knowledge used in the KM steps be closely related to the organisation's processes.

As stated before, the knowledge used by an organisation reflects its experience curve. A quick overview of an organisation's activities can reveal whether or not processes are completed. Only a deeper and more subjective view will capture the impact of mature members, who use knowledge learned through experience. The implication is that it is very difficult to demonstrate the value of KM. The outcomes generated by knowledge and experience are not typically included in traditional quantitative methods based on financial statements.

The problem of demonstrating the value of KM has been addressed from the perspective of LIS. This is not surprising, as library services are all KM processes after all. The LIS approach assumes that these hard-to-measure outcomes can be associated with an organisation's mission because they contribute to the organisation's goals. The next section discusses KM goals.

21.3.4.1 Knowledge Management Goals

For any organisational unit, goals are conceived to achieve an organisation's mission. Implementing and maintaining KM goals require change management (see Chap. 11, Development and Maintenance). For changing and maintaining a culture that is suitable for KM, the organisation must implement and enforce a series of goals.

The first and more general KM goal is to create an infrastructure for KM. Through the proper infrastructure, the remaining goals can be reached. KM first needs to determine what knowledge is to be managed. It then has to make it transparent to organisation's members who will use KM. They need to be educated on what knowledge needs to be managed.

KM needs to make available to an organisation's members the proper means for knowledge collection, providing them with proper training. The approach shall define how the knowledge is to be represented in order to guarantee better accessibility to it. For example, it is crucial that an organisation's processes (i.e., of the organisation's units) be included in the captured knowledge to ensure reuse. Knowledge quality is of utmost importance and KM must constantly conduct and maintain processes for validation and verification of knowledge. Note that all these measures require strong leadership support.

Following simple management principles, the entire process shall be monitored so as to guarantee quality and adherence to the chosen approach. Close monitoring of these goals will ensure compliance with the organisation's goals and with the fulfilment of its mission. A better view of KM is given by its cycle.

21.3.5 Knowledge Management Cycle

Like CBR, KM has a cycle too. The cycle varies depending upon the organisation it aims to serve. There is therefore a spectrum of cycles. The level of abstraction of proposed cycles may also vary, and so may the choice of words.

Figure 21.2 shows a minimal cycle on the left. On the right is a more comprehensive cycle. Note that they are, in essence, the same. The cycles may be interpreted as starting in create or capture knowledge, which can be preceded by reuse. They do not end; they continue perpetually, being triggered by organisation's members who, in aggregate, build the organisation's experience curve. One can also observe a relation to the CBR cycle, which we discuss next.

21.4 Case-Based Reasoning and Knowledge Management

We have been discussing KM; we now turn our attention back to CBR so we can compare them. CBR is a reasoning methodology that relies on recalling learned

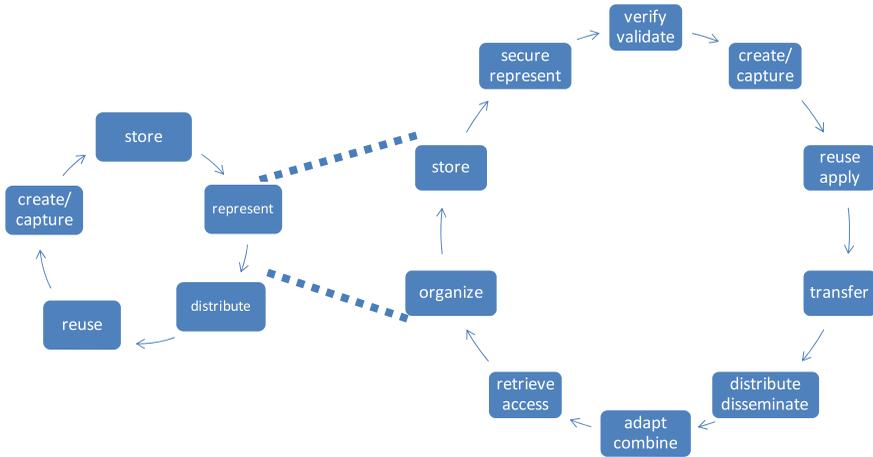


Fig. 21.2 A spectrum of knowledge management cycles

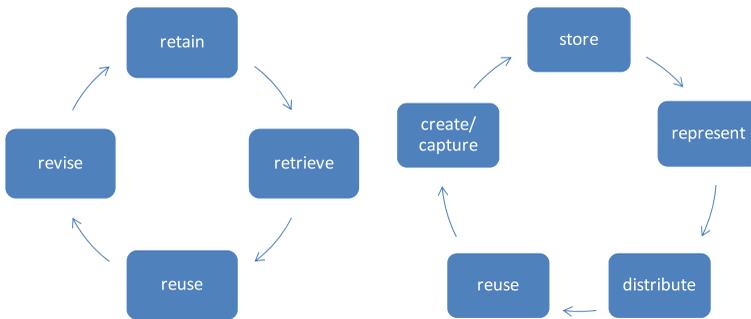


Fig. 21.3 CBR and KM cycles

and stored experiences and adapting them to solve new problems. KM is an organisational function that aims at embedding knowledge in processes in support of an organisation’s mission. It would be also accurate to describe CBR as a methodology that embeds knowledge to make decisions and solve problems. Furthermore, it would also be accurate to describe KM as a function that supports decision making by recalling existing experiences and adapting them to deliver organisational processes. In fact, they are both inherently the same concept, CBR emphasizing the computational aspects and KM the organisational functions. We may note that in KM the experiences need not be, and are usually not, represented as they are in CBR systems.

The affinities between CBR and KM become explicit as we compare both their cycles. Recall Chap. 2, Basic CBR Elements, where we introduced the CBR process model through a series of tasks, problem formulation, retrieve, reuse, revise, and retain. These tasks comprise the CBR cycle. Figure 21.3 shows both cycles.

As previously discussed, the cycles are quite similar. The CBR cycle describes the use of the CBR process while the KM cycle focuses on the processes of a KM organisational unit. The following section explains and illustrates the use of CBR methodology to implement KM cycles.

21.5 CBR Implementing KM Cycles

In this section we illustrate implementations of CBR systems that perform KM steps. Interestingly, as KM relates to knowledge, and all CBR tasks involve knowledge, it could be argued that all CBR implementations perform some form of KM. Nevertheless, if this were true then every intelligent or knowledge-based system could always be seen as a KM system. Therefore, we emphasize here that KM performs knowledge tasks that are processes of an organisation, and their results will likely impact how its goals and mission are achieved. Consequently, CBR should be noted in the KM context when a CBR implementation embeds an organisation's processes in any of its CBR knowledge containers.

21.5.1 *Knowledge Infrastructure and Organisation*

Code reuse is of great demand in software engineering. For this reason, there are many efforts to utilize CBR techniques for reuse of code in its various forms. The experience factory is one well-structured methodology that goes beyond code reuse. The experiences in the factory are not of the form used in CBR systems as pointed out in Sect. 11.4.3.

21.5.1.1 Experience Factory

When we first mentioned experience factory (EF) in Chap. 11, Development and Maintenance, we presented it as a tool to support development and maintenance of CBR systems. That was a perspective where EF can support CBR. In this chapter we discuss how CBR can be integrated into the EF for managing software engineering experiences to implement a KM cycle. Now CBR is providing support for the EF via KM.

This integration of CBR and EF depends on the fact that EF is an organisational framework for experiences whereas CBR has the techniques to implement reasoning tasks that involve experiences. Using CBR to implement knowledge tasks in the EF demonstrates an important and maybe not so obvious benefit of providing a computational infrastructure for a KM cycle. In this light, the EF is a KM approach.

The resulting integrated framework utilizes a series of experience bases. In this example, CBR is used to perform knowledge creation, analysis, representation, preservation, access, verification, validation, reuse and adaptation, and leveraging.

This resulting framework is sometimes referred to as an experience-based information system.

21.5.2 Knowledge Organisation and Retrieval

A well-known profession that specialises in organising knowledge for access is that of reference librarians. A CBR system can potentially realise this entire task. It is, however, not ideal to target a system that replaces humans because this would require the inclusion of several modules such as base ontologies for commonsense reasoning. A rational use of technology is one that complements humans in an integrated environment where the strengths of both computers and humans are maximally utilized. Consequently, the ideal use of a CBR system for knowledge organisation and access would complement the work of reference librarians by providing them assistance. This is a solution for reference librarians in three circumstances: (1) experienced librarians who struggle to keep up with the exponential growth of information resources; (2) novice librarians who are still gaining experience; and (3) one-person reference desks in small libraries, as in schools or law offices.

A CBR system to support reference services is based on its ability to organise information, incorporate expertise, and reuse and adapt previous successful answers.

The input to the system is exactly the same as that received by reference librarians, that is, reference questions (see the example in Table 21.1). The domain-independent nature of this task requires a categorization of case knowledge. This is easily done by grouping cases based on their domains. This generates the need for two-step retrieval, as discussed in Chap. 8, Retrieval. In the first step, a new reference question is used to identify its domain. In the second step, the reference question is used as a new problem. Cases are question-answer pairs.

Table 21.1 shows an example of a typical reference question and its original answer. These actual question-answer pairs are used to create the cases for the case base. Note that the answer includes resources searched by reference librarians. This is the valuable knowledge that can be shared with other reference librarians.

Question and answers like the one in Table 21.1 are fit to categories of questions, and then simplified for easier matching as a case. Each category shares similar resources for search. Table 21.2 shows examples of two cases originating from actual searches that can be shared among reference librarians.

These elements can be worked as case solutions in multiple ways. Some of the textual elements would need methods as discussed in Chap. 17, Textual CBR. The reuse of previous solution, requires adaptation methods, as discussed in Chap. 9, Adaptation.

With the goal of utilizing the strengths of both humans and computers, such a system would capture expertise embedded in a librarian's answers. Such expertise might otherwise never be explicitly stored. These domain-specific case bases could be potentially shared as cloud resources so librarians anywhere could access and reuse the expertise. This would give more time to reference librarians to use their intelligence to unravel hard cases, that is, searches that are novel and challeng-

Table 21.1 Question-answer example

Last Update:

2006-04-13 23:12:00.0

Question:

Patron needs to know what drugs use equine estrogens. She knows about Premarin. Are there others? Needs drug names and companies that manufacture them.

Answer:

We were only able to find two other drugs in the Physician's Desk Reference (PDR) in this category. They are named "Prempro" and "Premphase". They are described as (...) manufactures both of these is named Wyeth Pharmaceuticals, Philadelphia, PA.

In addition to the PDR, the Seattle Public Library has the following resources which cover this topic. These books are located on the 5th floor of the Central Library. 1. "Complete guide to prescription and non-prescription drugs". by HP Books, c1983- Call #: 615.1 C73865 2. "The Essential guide to prescription drugs". by Harper and Row, c1977- Call #: 615.1 L852E 3. "Prescription drugs" by the editors of Consumer guide. Call#: 615.1 P925

Keywords:

estrogen, horses, drugs, nonprescription

This material is from the QuestionPoint Global Knowledge Base.

Table 21.2 Category question and search sources

Question	Answer
Drugs (Other)	Physician's Desk Reference (PDR)
General (which drugs use equine estrogens)	Complete Guide to prescription and non-prescription drugs (book)
Statistics (death-rate) Treatment (Hepatitis C) Disease (Hepatitis C) General (Hepatitis C, infectious disease)	New York Public Library, Manhattan Branch (http://www.nypl.org/branch/central_units/mm/midman.html); Centre for Disease Control (FAQs); Gale Encyclopedia of Medicine; Medline Plus and HOAH; National Centre for Infectious Diseases; The Hepatitis Info Network, National Institute of Diabetes and Digestive & Kidney Diseases

ing. These unusual and challenging reference questions could also be shared among many human professionals who could use the precious time freed from repeated searches to find new solutions.

This use of CBR in support of reference librarians is also an example of the closeness of knowledge distribution and information retrieval. The association between CBR and information retrieval is later discussed in Chap. 23, Relations and Comparisons with Other Techniques.

21.5.3 Knowledge Retrieval and Reuse

Retrieval and distribution are inherently connected. Retrieval functions for CBR have been extensively discussed in Chaps. 8 and 14. The implication for KM is

in the distribution because it should be made in a way that motivates other KM processes (e.g., sharing, leveraging, and reuse).

One of the main recommendations in knowledge distribution is to present the knowledge to a potential user when and where it is needed. This implies that KM approaches should be embedded in the environment of the users rather than create new standalone tools for knowledge distribution.

There are a variety of modes for knowledge distribution. The two main categories are passive and active. Passive modes require the user's initiative whereas in active modes knowledge is distributed without the user's request. Another dimension of distribution refers to the number users receiving it; distribution can be broadcast or personalized. Based on the principle mentioned above, the ideal is the active move with personalized distribution delivered in the context of the process for which it is needed, that is, where and when it can be reused.

21.5.3.1 Knowledge Reuse

A properly designed infrastructure for knowledge organisation, distribution and access is the main benefit of supporting knowledge transfer. This support of knowledge transfer is crucial in all fields, so an organisation's members can make the right decisions that will help their organisation achieve its mission and prevent undesired consequences. In some domains the mission of these organisations involves critical aspects like well-being, order, and safety, whereas undesired consequences may include loss of lives. Such organisations are, for instance, dedicated to healthcare and security. Next, we describe an application of CBR for knowledge transfer and reuse for such organisations.

Most organisations today collect lessons learned. One of the early adopters are governmental organisations with members in thousands that use advanced technologies. Examples are space agencies and military organisations. A substantial part of the work in such organisations is in simulated exercises, after which members are asked to describe what they learned and store this information in lessons learned systems.

Lessons learned are described and captured in a variety of forms. A complete lessons learned must include a series of contents, namely, the learned strategy, how it was learned, and how it is applicable for reuse. Extensive work has been done on delineating attributes that characterise high-quality repositories. For reuse, it is essential that these contents be included.

Just as there are many ways in which lessons learned can be distributed, there are also different ways they can be reused. In general terms, when processes or operations are to be delivered manually by humans, lessons must be presented to these humans *at the time and in the context* of the processes for which lessons are applicable.

Consider a KM approach that uses CBR integrated into a system for planning operations. The plan is designed one task at a time. The case-based KM system can track each task included in the plan and search its repository for lessons that are applicable to each task.

Table 21.3 New query interpreted by the case-based lessons learned system

Target process:	transport supplies to affected area
Specific contextual indices:	disaster relief

Table 21.4 Knowledge unit in form of a case

Target process:	transport supplies to affected area
Specific contextual indices:	disaster relief
Lesson strategy:	include disaster medical supplies, that is, those that direct limited resources to the greatest number of individuals (as opposed to emergency medical supplies that direct maximal resources to a small number of individuals)

Suppose an operation is needed for disaster relief. The plan being created includes tasks to bring personnel and medical supplies and to rescue the wounded. The step in the plan is a target operation process, “transport supplies to affected area”. This is the most important index used by the case-based lessons learned system. This is what we mean by embedding the target process in the knowledge unit. There must be a way that the KM system can recognise these processes to match them against its knowledge repository whenever they are about to be executed by an organisation’s member. The case-based KM system creates the new query as in Table 21.3.

Table 21.4 shows the case that was retrieved when the query was submitted. Note that the requirement that knowledge has to be distributed when and where it is useful poses high demands on the similarity threshold. Most or all the indices should match for a case to be distributed. This type of proactive distribution requires caution, as useless interruptions are not tolerated.

This is a simple example of a lesson that changes a facet of a plan based on a specificity that needs to be made available to the user in charge of planning the operation *at the time and in the context* in which it was needed, where it is applicable.

Though it may seem that the lesson content is obvious, when considering the multiple nature of operations that users may be subject to, lessons targeting specificities of each operation context are extremely important. The automated inclusion of lessons learned in simulated operations has been shown to reduce casualties significantly.

21.5.4 Knowledge Sharing

Knowledge sharing may be the most popular KM task. There are two very distinct reasons why knowledge sharing is challenging. The first is due to the nature of people, the second is due to the nature of knowledge.

Knowledge sharing within oneself is the simple remembering of an episode. Once you learn how to better perform a task, you usually remember that. But even the closest human to you will not benefit from the knowledge you have unless

you explicitly share it. Knowledge sharing thus is only an issue when more than one member has the potential to reuse the knowledge. Knowledge sharing requires awareness of the possession of the knowledge, complete lack of barriers for sharing, understanding the knowledge needs of others, and opportunity for sharing.

Knowledge can be of a heterogeneous character and will usually have different sources that can be in conflict. Consider an example. Suppose a company can produce a product in two different ways, one that is environmentally friendly and the other that is not but is cheaper to produce. For both environmental friendliness and economic feasibility, the same organisation may have members who will give conflicting recommendations based on the goals of their departments. A final decision has to be made that combines knowledge shared by both.

21.5.4.1 Sharing and Leveraging in Science Collaboratories

Collaboratories are virtual organisations that aggregate individuals working with a scientific purpose. When the users of a KM system are scientists and engineers who produce scientific knowledge, then a KM cycle can support knowledge representation, sharing, transfer, and leveraging.

The main challenge in collaboratories is the vocabulary and the adoption of an agreed format for knowledge representation. Ways to promote knowledge sharing include easy visualisation of colleagues' works, associations between works that have a methodological or topic overlap, and active distribution of knowledge. Above all, interfaces should be simple and contents minimal but sufficient so others can recognise the potential of collaboration.

As previously discussed, as with other KM efforts, demonstrating results is always a challenge. In scientific collaboratories, however, once scientists share their findings, they can also be asked to relate their work to that of others. Those relations when informed by scientists are evidence of knowledge sharing. At times, scientists will explicitly indicate an association between two knowledge units where the latter leverages knowledge of the former.

The contribution of CBR to collaboratories is multifold. Initially, CBR can guide the format of knowledge units that are contributed, linked, and shared. CBR can support a problem-oriented retrieval for search and for opportunities for active knowledge distribution.

The CBR guidance on formatting scientific contributions is to represent them as problem-solution pairs. Usually, KM approaches encourage users to share knowledge once it is learned. For scientific communities, sharing should occur before the process is completed and the novel scientific contribution is learned. This is because sharing motivations and ongoing research questions or hypotheses encourages collaboration. Sharing only after the fact will encourage knowledge leveraging but not collaboration (i.e., at least the opportunity to collaborate on that specific effort has passed). The knowledge format we demonstrate in Table 21.5 has three temporal dimensions for problem-solution cases. Note that the shaded last two rows represent the problem, that is, indexing elements; the two first rows are the solution, the reuse elements.

Table 21.5 Format for scientific contributions

Prior	Ongoing	Completed
State what is known and what needs to be learned	Declare what you are trying to learn, hypotheses	State what you learned
State the support for this	State what will be done to learn it, experimental design	State the support, your results
Where this knowledge is applicable	Explain its usefulness, where this is applicable	Task or process for which this knowledge is applicable
More specific details	More specific details	More specific details

Now we give an example of a prior and a completed unit that reveals another benefit of this format. It is both an example and a statement about this representation. For prior motivation, “it is hard to motivate users to contribute to KM systems”. As support, many references can be used. For indexing elements of the prior case, “This is knowledge applicable in designing, developing, and deploying KM systems”. The specifics are, “KM system is of the type repository-based”. A completed unit would be for what was learned: “by using a structured representation format like the one in Table 21.5, it becomes easier to generate reports about knowledge entered in the KM systems”. For completed support we would have, “we learned that providing reports to the users is a motivation for their use of the system”.

21.6 For Which KM Tasks Should I Use CBR?

Service organisations like consulting businesses realise their mission through the work of people. Regardless of whether service employees are of high or low level of qualification, there is a lot of knowledge they use that is kept in their minds. This is a situation where building a CBR system to capture and store experiential knowledge can be beneficial.

In Chap. 12, Advanced CBR Elements, we discussed contexts and distinguished different levels. We stated that CBR favour a group level. Often, such a level is given when one considers KM in companies.

A well-designed interface for capturing experiences is crucial for success. All goals discussed in Sects. 21.3.4.1 need to be in place. Cases need to be represented as problem-solution pairs, where one index to guide retrieval is the target process for which experiential knowledge is applicable. The similarity will give a high weight for this attribute, while determining its applicability with specific features that will discriminate applicable situations. For example, an experience only applicable in the evening may have been learned; thus we do not want this to be distributed if the process is to be delivered during the day.

Before starting a new project, the employees themselves or their supervisors can search the system for applicable experiential knowledge. This is also an opportunity for knowledge capture. The reuse step after distribution (i.e., retrieval) should allow

for adaptation, i.e., for users to enter new experiences they are reminded of when using the system. CBR methodology entails knowledge adaptation as an essential element of its underlying methodology. CBR is based on the notion of reusing and adapting previous experiences. Adaptation methods are discussed in Chap. 9. Adaptation can also lead to knowledge creation. Additionally, a thorough analysis of previous knowledge may reveal gaps that can guide new simulations to learning missing knowledge.

The importance of adapting knowledge is significant. Consider the cycle of knowledge reuse implemented by humans obtaining knowledge from experts to search (i.e., for experts), negotiate (i.e., contact, schedule meetings), retrieve (i.e., elicit knowledge from experts), and adapt. Studies show that the effort allocated for adapting knowledge is greater than the effort needed for the three other individual steps.

It is therefore expected that the improved ability to automate knowledge adaptation will be more beneficial than improvements to search or distribution. This speculation is supported by the current level of sophistication that search methods have reached; it seems that new improvements in search would produce increasingly smaller benefits, while even a small improvement in adaptation may lead to significant benefits.

If human experts, who have learned through experience, are not available, then experiential knowledge might be available in documents. If documents entail problem-solution pairs that contain useful and reusable knowledge, one possibility is to rely on textual CBR methods, discussed in Chap. 17, to create a CBR system to reuse such knowledge.

It may also be the case that previous episodes of problem solving are recorded in structured databases with distinct fields. Consider examining if these fields individually retain problem and solution values. Then, the next step is to conceive a similarity measure and use those records as cases.

Examples in Sect. 21.5 illustrate many ways in which CBR is used to implement KM tasks. These implementations illustrate how CBR can be used as a computational methodology for KM. Furthermore, they have two other benefits. One is to document an organisation's intellectual assets. The other is that, as processes are embedded in the stored knowledge, they can be easily associated with an organisation's mission, making the implementation also an instrument to demonstrate the value of KM.

21.7 Tools

All CBR tools previously mentioned in previous chapters (e.g., CBRWorks, Orange, jColibri) can be used for knowledge management tasks. Other tools available for knowledge management (e.g., Microsoft® Sharepoint) are widely used in organisations for collaboration and content and document management. These tools, however, are not designed to perform KM tasks themselves; they do not include CBR. The human users of those tools have to perform the reasoning tasks themselves.

21.8 Chapter Summary

Knowledge management (KM) concerns the proper allocation, coordination and planning of an organisation's intellectual assets. Despite being an organisational problem, its solutions span multiple disciplines. The perspective adopted here is that of information technology, using the CBR methodology to perform KM tasks.

The information technology perspective of KM recognises knowledge as what enables decision making and problem solving. From this view, multiple organisational processes in virtually every domain can benefit from KM. KM embeds knowledge for quality decision making while members deliver organisational processes.

One of the main challenges of KM is demonstrating its effectiveness. Because it aims at supporting organisational processes, and these processes can be associated with an organisation's mission, knowledge units can be linked to an organisation's mission. Explicitly representing them in a CBR system will help document KM steps, its efforts, uses, results, and impact.

The relation between KM and CBR is intrinsic. Their cycles reveal that both have manipulation and reuse of knowledge at the core. Consequently, CBR can be used to perform multiple KM steps. This chapter concludes by describing which KM tasks can be implemented with CBR.

21.9 Background Information

A review of the work done through 2005 is discussed in Althoff and Weber (2006).

The six definitions of knowledge are given in Alavi and Leidner (2001). (1) Knowledge can be considered a process that applies expertise. (2) Knowledge is a capability because it can alter the outcome of a process. (3) Knowledge is an object and thus can be manipulated. (4) Knowledge can be a condition when it accesses information. (5) Knowledge is related to understanding and thus it is related to learning. (6) Knowledge can also be described as information that is tailored to a particular individual or situation.

The simplified decision making and problem solving model was proposed by Huber (1980). The problem of the relation between different challenges for KM was investigated in the IMCOD project; see Bachmann and Dridi (1994).

The difficulty of demonstrating the value of any KM effort was addressed in Abels et al. (2002, 2004). They explain that outcomes generated by knowledge and experience are not suitable for traditional quantitative methods based on financial statements. They identify and measure the value of library services by associating them to the organisation's mission because they contribute to the organisation's goals—the CLIS method. Examples of those library services are timely support for decision making and for the development of policies.

The knowledge cycle presented in Fig. 21.2 with only four steps was suggested by Weber and Kaplan (2003) as a conceptual cycle. It suggests that these steps are always represented in every proposed knowledge cycle and can entail all variations of the tasks.

The goals we discuss that an organisation must meet for successful KM were devised by Marshall et al. (1996). They studied the relationship between management success and financial health through the intrinsic relationship between risk management and knowledge management. These authors strongly support the use of technology for implementing KM tasks. They argue that, to be truly effective, KM requires organisational change, making the organisation responsible for directing the change by implementing and enforcing a series of KM goals.

The association between the CBR and KM cycles has been discussed in the literature before (e.g., Watson 2003). In his book, the minimal cycle is described through four steps called acquire, analyse, preserve, and use knowledge.

Aamodt and Plaza (1994) introduce the CBR cycle and names the four R's in the cycle, retrieve, reuse, revise, and retain, which was discussed in Chap. 2, Basic CBR Elements.

A CBR system to complement the work of reference librarians along the lines we described in Sect. 21.5.2, including its tables, was proposed by Bui (2007), who utilizes examples from QuestionPoint—an online reference service made available to the public courtesy of the New York Public Library. See <http://www.questionpoint.org/crs/servlet/org.oclc.home.BuildPage>.

A review of knowledge distribution is given in Weber et al. (2001). The review categorizes, describes and exemplifies multiple modes and their uses in KM systems.

Barriers to knowledge sharing have been extensively discussed in multiple publications. See, for example, Weber (2007), Disterer (2001), and Szulanski (1996).

Science laboratories have been defined in Wulf (1993) and described in Finholt and Olson (1997). Weber et al. (2006, 2008) describe the development of a KM approach to support scientists where knowledge sharing is demonstrated via associations entered by scientists. Weber et al. (2008) describes the use of the format exemplified in Table 21.5.

Jacobson and Prusak (2006) present the results of a study describing the proportion of effort allocated to different knowledge tasks. Their results are as follows: “Searching for knowledge, 10.2 %; scheduling meetings with experts, 6.2 %; eliciting knowledge from experts, 37.7 %; adapting knowledge gained, 45.9 %”. They conclude that the future payoffs will be on strategies that facilitate knowledge adaptation.

The experience factory (EF) model (Basili 1995) is an organisational approach for continuously learning from experience. Therefore, CBR is an obvious implementation technology for an EF (Henninger 1995). It has been integrated with CBR by these authors: Althoff and Wilke (1997), Tautz and Althoff (1997), and Althoff et al. (1998).

Weber et al. (2001) describe and illustrate the potential positive consequences of adopting the CBR methodology. This article integrates ideas collected from the AAAI Intelligent Lessons Learned Systems Workshop (Aha and Weber 2000). For the practical adoption of CBR as the underlying framework for lessons learned, Weber et al. (2001) propose a case representation for lessons learned. The case representation was later used in the monitored distribution (MD) approach for proactive

distribution of lessons learned (Aha et al. 2001). A description of lessons learned includes the organisational process that it targets. Therefore, MD can be integrated with organisational systems. MD motivates the reuse of a knowledge artefact by bringing to the attention of the user when and where it is applicable and by including a rationale for its reuse (Weber and Aha 2003). The benefit of the MD approach has been demonstrated in an experiment that simulates military operations planned with and without the reuse of lessons learned, taken from the NLLS (Navy Lessons Learned System) repository.

21.10 Exercises

Exercise 1 Describe a KM task that is seamlessly performed internally by individuals.

Exercise 2 How are the KM and CBR cycles distinguished?

Exercise 3 A user submits a knowledge artefact to a KM system and creates a link that associates this new artefact with a previous one and labels the association “uses”. What kind of KM task was performed by the user while creating the new artefact?

sharing leveraging creating associating

Exercise 4 Identify a KM task you are familiar with currently being performed by humans.

Exercise 5 Identify a KM task you are familiar with currently being performed by a computer system.

Exercise 6 What kind of system would you recommend to a KM task you are familiar with currently being performed by humans.

Exercise 7 An industry recently replaced some production line workers with robots. Instead of firing the workers, the management trained them to observe the quality of the produced parts so they could indicate when wear and tear required the robots to be removed for maintenance. The problem was that when the workers realise that parts were coming out with defects, too many had already been produced, causing excessive parts to be rejected. What kind of KM approach would you propose that could potentially decrease the volume of rejected parts?

References

Aamodt A, Plaza E (1994) Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Commun* 7(1):39–59

- Abels EG, Cogdill KW, Zach L (2002) The contributions of library and information services to hospitals and academic health sciences centers: a preliminary taxonomy. *J Med Libr Assoc* 90(3):276–284
- Abels EG, Cogdill KW, Zach L (2004) Identifying and communicating the contributions of library and information services in hospitals and academic health sciences centres. *J Med Libr Assoc* 92(1):46–55
- Aha DW, Weber RO (eds) (2000) *Intelligent lessons learned systems: papers from the AAAI 2000 workshop*. Technical report WS-00-03. AAAI Press, Menlo Park
- Aha DW, Weber RO, Muñoz-Avila H et al (2001) Bridging the lesson distribution gap. In: *IJCAI 2001: seventeenth international joint conference on artificial intelligence*, vol 2, Seattle, WA, 2001. Morgan Kaufmann, San Francisco, p 987
- Alavi M, Leidner D (2001) Review: knowledge management and knowledge management systems: conceptual foundations and research issues. *MIS Q* 25(1):107–136
- Althoff K-D, Weber RO (2006) Knowledge management in case-based reasoning. *Knowl Eng Rev* 20(3):305–310
- Althoff K-D, Wilke W (1997) Potential uses of case-based reasoning in experience based construction of software systems and business process support. In: Bergmann R, Wilke W (eds) *GWCBR'97: sixth German workshop on case-based reasoning: foundations, systems, and applications*. Technical report LSA-97-01E, centre for learning systems and applications. University of Kaiserslautern, p 31
- Althoff K-D, Birk A, Gresse von Wangenheim C, Tautz C (1998) Case-based reasoning for experimental software engineering. In: Lenz M, Bartsch-Spörl B, Burkhard H-D et al (eds) *Case-based reasoning technology: from foundations to applications*. Lecture notes in artificial intelligence, vol 1400. Springer, Berlin, p 235
- Bachmann B, Dridi F (1994) Definition of a communication layer for expert systems. In: Chen JG (ed) *6th international conference on AI and expert system applications*, Houston, Texas, 1994
- Basili VR (1995) The experience factory and its relationship to other quality approaches. *Adv Comput* 41:65–82
- Bui Y (2007) Case-based support for library reference services. In: Weber RO, Richter MM (eds) *ICCBR 2007: case-based reasoning research and development*. 7th international conference on case-based reasoning, Belfast, UK, August 2007. Lecture notes in computer science (lecture notes in artificial intelligence), vol 4626. Springer, Berlin, p 507
- Disterer G (2001) Individual and social barriers to knowledge transfer. In: *HICSS 2001: 34th annual Hawaii international conference on system sciences*. IEEE, Los Alamitos
- Finholt T, Olson G (1997) From laboratories to laboratories: a new organizational form for scientific collaboration. *Psychol Sci* 8(1):28–36
- Henninger S (1995) Developing domain knowledge through the reuse of project experiences. In: Samadzadeh MH, Zand MK (eds) *SSR'95: ACM SIGSOFT symposium on software reusability*, Seattle, WA, 23–30 April 1995
- Huber GP (1980) *Managerial decision making*. Scott, Foresman, Glenview
- Jacobson A, Prusak L (2006) The cost of knowledge. *Harv Bus Rev* 84(11):34
- Marshall C, Prusak L, Shpilberg D (1996) Financial risk and the need for superior knowledge management. *Calif Manag Rev* 38(3):77–101
- Szulanski G (1996) Exploring internal stickiness: impediments to the transfer of best practice within firms. *Strateg Manag J* 17(winter):27–44
- Tautz C, Althoff K-D (1997) Using case-based reasoning for reusing software knowledge. In: Leake DB, Plaza E (eds) *ICCBR 1997: case-based reasoning research and development*. Second international conference on case-based reasoning, Providence, RI, July 1997. Lecture notes in computer science (lecture notes in artificial intelligence), vol 1266. Springer, Berlin, p 156
- Watson ID (ed) (2003) *Applying knowledge management: techniques for building corporate memories*. Morgan Kaufmann, San Francisco
- Weber RO (2007) Addressing failure factors in knowledge management. *EJKM* 5(3):333–346
- Weber RO, Aha DW (2003) Intelligent delivery of military lessons learned. *Decis Support Syst* 34(3):287–304

- Weber RO, Kaplan RM (2003) Knowledge-based knowledge management. In: Jain R, Abraham A, Faucher C, Zwaag BJ (eds) *Innovations in knowledge engineering*. International series on advanced intelligence, vol 4. Advanced Knowledge International, Adelaide, pp 151–172
- Weber RO, Aha DW, Becerra-Fernandez I (2001) Intelligent lessons learned systems. *Int J Expert Syst Res Appl* 20(1):17–34
- Weber RO, Morelli ML, Atwood ME et al (2006) Designing a knowledge management approach for the CAMRA community of science. In: Reimer U, Karagiannis D (eds) *PAKM 2006: practical aspects of knowledge management*. 6th international conference, Vienna, Austria, November/December 2006. *Lecture notes in computer science (lecture notes in artificial intelligence)*, vol 4333. Springer, Berlin, p 315
- Weber RO, Gunawardena S, Abraham G (2008) Representing and retrieving knowledge artifacts. In: Yamaguchi T (ed) *PAKM 2008: practical aspects of knowledge management*. 7th international conference, Yokohama, Japan, November 2008. *Lecture notes in computer science (lecture notes in artificial intelligence)*, vol 5345. Springer, Berlin, p 86
- Wulf WA (1993) The collaborative opportunity. *Science* 261(5123):854–855

Part V

Additions

The chapters may be of interest to readers who want additional information about formal aspects and areas that are related to CBR. They are not required for understanding the previous parts.

Chapter 22, Basic Formal Definitions and Methods, deals with formal aspects of three contexts discussed in the book. They are logic and truth, information content, and utility.

Chapter 23, Relations and Comparisons with Other Techniques, is concerned with areas that have a close relation to CBR. Some of these areas can be seen as alternative or complementary to CBR. These are database management systems, information retrieval methods, pattern recognition, knowledge-based systems, and machine learning. Other methods have influenced CBR. These are methods for the treatment of uncertainty, and the areas of cognitive science, knowledge management, and again machine learning.

For these areas we exhibit commonalities as well as distinguishing factors. We also support the reader who has the choice:

- Should I use CBR or something else?
- Which additional methods should I look at to increase the power of a CBR system?

Many of these questions have been considered throughout the book; here, we give a systematic overview.

Chapter 22

Basic Formal Definitions and Methods

22.1 About This Chapter

This chapter contains additional information about formal aspects mentioned earlier in the book. These are threefold: (a) Is the provided information correct? (b) How much information is provided? (c) Is the information useful? In this chapter we provide the formalisms for these views. This chapter is not intended to be used as a textbook by students for learning these topics. It is intended as a support for this book's readers. It discusses formal aspects in the adopted terminology, which is often varying in the literature. As previously mentioned, this chapter does not have a tools section.

22.2 General Aspects

This chapter discusses precise formulations, used in certain computational applications. Computational implementations are formal systems by their very nature. When implementing them, one should always be aware of the fact that formal systems model reality. The art of modelling is to define the formally obtained context in such a way that an intended problem can be tackled by a computer.

For a given package of knowledge there are three central views:

- The logic view: Are this knowledge and the reasoning correct?
- The information-theoretic view: How much information is contained in the package?
- The utility view: Is this knowledge useful?

In this chapter we discuss the three aspects insofar as they are of interest to CBR.

22.3 Correctness and Logic

First, we state that logic is not the same as a logical formalism. People can argue logically without referring to a formal method. In fact, this is almost always the case, even when talking about mathematics. Only formal logic consists of formalisms. On the other hand, it is not certain that true formal statements are always of interest. In fact, they can contain complete nonsense.

Often one needs preciseness in the sense of formalisms, in particular, in the context of an implementation. This preciseness is obtained by abstracting from imprecise elements. The formalism has two parts:

- A syntax that defines what the well-formed expressions of the language are;
- A semantics that defines the meanings of the expressions.

It was one of the great achievements of science when Aristotle formalized (a part of) logic, known as classical logic. The popularity of classical logic arises from the fact that it combines wide expressive power with nice formal properties. It provides the most commonly used formalisms for exact representation. One distinguishes between propositional and predicate logic. The former provides the simplest model for logic. Predicate logic is an extension. Classical logic and its fragments are built according to the local-global principle. In fact, it made the principle popular without carrying this name.

22.3.1 Propositional Logic

In the syntax of propositional logic we have as basic (atomic) symbols:

- (a) Propositions: Symbols such as p , r , and so on, and the particular propositions TRUE and FALSE. The set of propositions is referred to as Prop.
- (b) Logical (Boolean) connectives: \wedge (and, conjunction), \vee (or, disjunction), \Rightarrow (implication, conditional, If-Then), \Leftrightarrow (equivalence, biconditional), \neg (negation).
- (c) In addition, for grouping, parentheses are used.

Now we can define formulas, which are expressions of propositional logic. Boolean connectives join propositions together into complex formulas.

Definition 22.1

- (1) True, False or any proposition symbol, p , q , r , is a formula; these formulas are also called *atomic*.
- (2) If ϕ and ψ are formulas, then $\phi \wedge \psi$, $\phi \vee \psi$, $\phi \Rightarrow \psi$, $\phi \Leftrightarrow \psi$ and $\neg\phi$ are formulas.
- (3) Each formula is built in this way. The set of all formulas is denoted by *Form*.

The meaning of propositional formulas is rather restricted. The semantics of propositional logic knows only the truth values 0 (for false) and 1 (for true). That means that in a communication the message “ p ” makes sense only if the partners have already agreed on what p means.

Formally, the semantics is given by a truth evaluation of the formulas.

Definition 22.2

- (1) An assignment is a mapping $\text{ass}: \text{Prop} \rightarrow \{0, 1\}$ such that $\text{ass}(\text{True}) = 1$ and $\text{ass}(\text{False}) = 0$.
- (2) The assignment is extended to all formulas according to the well-known rules:
 - a. $\text{ass}(\phi \wedge \psi) = 1$ if and only if $\text{ass}(\phi) = 1$ and $\text{ass}(\psi) = 1$;
 - b. $\text{ass}(\phi \vee \psi) = 1$ if and only if at least one of $\text{ass}(\phi)$ and $\text{ass}(\psi)$ has value 1;
 - c. $\text{ass}(\phi \Rightarrow \psi) = 0$ if and only if $\text{ass}(\phi) = 1$ and $\text{ass}(\psi) = 0$;
 - d. $\text{ass}(\phi \Leftrightarrow \psi) = 1$ if and only if $\text{ass}(\phi)$ and $\text{ass}(\psi)$ have the same value;
 - e. $\text{ass}(\neg\phi) = 1$ if and only if $\text{ass}(\phi) = 0$.

This means that we can compute the truth value of a formula once we have assigned a truth value to its atomic parts.

22.3.2 Predicate Logic

Predicate logic extends propositional logic substantially. The idea is not only to allow more expressions but also to extend the semantics beyond the mere truth values in propositional logic.

In the first place, there are more syntactic elements:

- Function symbols: $f, g, \dots, f_1, g_1, \dots$; each function symbol has an arity that is an integer.
- Predicate symbols: $P, Q, \dots, P_1, Q_1, \dots$, that also have an integer as arity.
- Variables: $x, y, z, \dots, x_1, y_1, z_1, \dots$
- Constants: a, b, c, \dots
- Additional logical symbols \forall and \exists (universal and existential quantifiers).

Next, we define terms as generalizations of constants, and this enables us to define the formulas.

Definition 22.3

- (1) Each constant and variable is a term.
- (2) If f is an n -ary function symbol and t_1, \dots, t_n are terms then $f(t_1, \dots, t_n)$ is a term.
- (3) All terms are generated from (1) using (2).

This allows us to define formulas as the legal expressions:

Definition 22.4

- (1) If P is an n -ary predicate symbol and t_1, \dots, t_n are terms then $P(t_1, \dots, t_n)$ is an atomic formula; each atomic formula is a formula.
- (2) If φ and ψ are formulas then $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $\varphi \Rightarrow \psi$, $(\varphi \Leftrightarrow \psi)$ and $(\neg\varphi)$ are also formulas.
- (3) If x is a variable and ϕ a formula then $\forall x(\varphi)$, $\exists x(\varphi)$ are formulas. The *scope* of \forall and \exists is the formula φ . A variable is called free if it is not in the scope of a quantifier.
- (4) All formulas are obtained from (1) by iteration using (2) and (3).

$\varphi(x_1, \dots, x_n)$ indicates that the variables are free in the formula, i.e., are not in the scope of any quantifier.

The semantics of formulas determines the meaning of the formulas. This was quite simple for propositional logic and is considerably extended for predicate logic; it is defined with respect to models. Models are the domains of interest, i.e., the domains we are talking about. In the language we have symbols only, but in a model function and predicate symbols obtain an interpretation, i.e., a meaning.

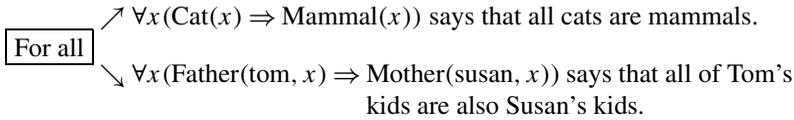
Definition 22.5

- (1) A model is of the form (U, R, F) , where U is a set (called the universe of the model), R is a set of n -ary relations and F is a set n -ary functions over U .
- (2) An interpretation of a predicate logic language is a mapping I that assigns
 - (a) To each predicate a relation, preserving the arity.
 - (b) To each function symbol a function, preserving the arity.
 - (c) To each constant an element of U .
- (3) A variable assignment is a mapping v of the variables to elements of U . Such an assignment is inductively extended to all terms.
- (4) A variable assignment v satisfies an atomic formula $P(t_1, \dots, t_n)$ if the n -tuple of the assignment of the terms is in the relation R , where $R = I(P)$ is the interpretation of P .
- (5) For Boolean connectives the satisfaction is defined in the same way as in the truth definition for propositional formulas in the above definition.
- (6) A variable assignment v satisfies $\forall x(\varphi)$ (or $\exists x(\varphi)$) if all (or some) assignments v' that agree with v (except possibly x) satisfy φ .

We remark here that although we might ultimately be interested in truth only, we have to consider satisfiability too. The reason is the existence of variables. For instance, $2 + 2 = x$ is neither true nor false if we do not know what x is.

An example of a function is $\text{Father}(x)$. One should observe here that the difference between “Father” as a relation and a father as a function is that the function assigns to each person its father; $\text{Father}(x, y)$ is a relation between two persons, indicating that x is the father of y .

The intention of the meaning of the quantifiers is illustrated by an example:



$\forall x \forall y (\text{Married}(x, y) \Rightarrow \text{Married}(y, x))$ expresses that the relation “Married” is symmetric.

Existential quantification is done in the same way.

We say a formula is true (false) in a model if it is satisfied by all (no) assignments. As a consequence, a formula may be neither true nor false in a model. A tautology is a formula that is true under any interpretation.

Continuing the example, suppose we have the predicate “ x is male” and $F(x, y)$ means “ x is the father of y ”, and suppose we have as assumptions:

“all children of x are male” and “ x is the father of a son”

Can we now conclude?

$\Gamma(x) : \text{“}x \text{ has only sons as children”}$

No! For two reasons:

- (a) The system is not informed about any relation between sons and children.
- (b) The system is not informed about any relation between male and son.

That means the system can only deal with knowledge that is formally represented and not with something that is in the head of the representing human. But on a formal level, the system is absolutely sure.

In predicate logic one describes properties by predicates. In Chap. 2, Basic CBR Elements, we introduced attribute-value representations as a major representation method. Now we will briefly point out the relations between the two.

The following example illustrates how easy the translation to an attribute representation is. Suppose we have an object represented by some constant a and two colours red and blue, both represented by a predicate with the intention:

If the object is red, then $\text{Red}(a)$ is true.

This representation becomes rather clumsy if we have many colours. In an attribute-value representation we can introduce an attribute Colour. This is a function that assigns to each object its colour. Hence such an attribute has a value set consisting of all colours, red, blue and others.

This vector fully represents the object a . The comparison between predicates and attributes is summarized in Table 22.1. The objects are the elements of the universe one is talking about as shown in Table 22.1.

In predicate logic one considers one element of the universe at a time and assigns truth values to the elements and the properties while attributes assign to each object the property that holds for the object. Next, we discuss some special and frequently used classes of formulas in the CBR context.

Table 22.1 Properties versus attributes

Attributes	Property representation
Predicate P represents a property	Property $P = \text{true}$ for object a
Attribute A assigns a property P to object a	$A(a) = P(a)$ for object a

22.3.3 Constraints

A constraint is simply a formula $\varphi(x_1, \dots, x_n)$ with some free variables x_i , and a constraint system is a finite set of constraints read as a conjunction. The presentation of a constraint system is called a “constraint satisfaction problem” and a solution in a model is a variable assignment that simultaneously satisfies all constraints, i.e., it satisfies their conjunction. A constraint restricts the possible assignments of its variables. If these variables occur in other formulas their assignments are restricted there too. This process is called constraint propagation.

Constraints occur regularly as preconditions of actions, for instance, in the context of adaptation. They are also used to exclude unwanted or impossible suggestions for solutions to problems. This takes place if constraints are violated. Sometimes the constraint system is overspecified, i.e., no solution exists. Then the system has to be relaxed, meaning that some constraints are deleted. This can be done by various methods and leads to the concept of weak constraints.

Definition 22.6 A weak constraint is one that one wants to be satisfied if possible.

If there are several weak constraints, one can arrange them in different ways, for instance:

- (a) Assigning relevancies and keeping only the most relevant ones. This corresponds to the weighting of attributes.
- (b) Arranging the constraints in a hierarchy and satisfying them top-down.

As an example, one can define the relevance using the costs that arise if the constraints are violated.

This leads into the area of uncertainty and is closely related to dealing with weight factors for linear similarity measures.

22.3.4 Rules

In logic, the term rule is used in different ways.

- (1) Rules as special formulas, namely, logical implications: $\phi_1 \wedge \phi_2 \wedge \dots \wedge \phi_n \Rightarrow \psi$.

The ϕ_k are the preconditions of the rule and ψ is the conclusion.

- (2) Rules of inference: They are used for inferring logical consequences of given formulas. Rules of inference are not elements of the language but means to derive formulas from assumptions.

Both are the basis of reasoning in logic, which we discuss in the next section. Important extensions are rules where the conclusion is an action that has to be executed. This is again central for adaptation.

22.3.5 Reasoning in Logic

The purpose of reasoning is to derive new information from a given one. The methods for this are logical rules or rules of inference. A general rule is of the form

$$\frac{\phi_1, \phi_2, \dots, \phi_n}{\psi}.$$

The ϕ_i are the premises of the rules and ψ is its conclusion.

For each such rule it is required that the truth (or satisfaction) of the premises ensures the truth (satisfaction) of the conclusion. This makes the application of the rules reliable.

Two important inference rules are:

- (1) The rule “modus ponens” that combines the rules as formulas with inference:

$$\frac{\phi, \phi \Rightarrow \psi}{\psi}.$$

- (2) The universal quantification rule that expresses the intuitive idea of “for all”:

$$\frac{\forall x \phi(x)}{\phi(a)}.$$

Here a is an arbitrary constant in the language. One should observe that the opposite rule, namely, inferring $\forall x \phi(x)$ from the validity of $\phi(a)$ for all constants a , is not valid. The reason is that not all elements may be denoted by a constant.

Inference rules give rise to formal proofs (deductions) as the steps of reasoning.

Definition 22.7 A deduction of a formula φ from a set of formulas Σ is a sequence $(\varphi_1, \varphi_2, \dots, \varphi_n)$ such that

- (1) Each φ_j is either in Σ or a consequence of some formulas φ_k where $k < j$.
- (2) $\varphi = \varphi_n$.

The attribute-value representation itself has no internal reasoning methods. However, reasoning using outside methods can take place in three ways:

- (1) Translate the representation into predicates and use their inferences. This is truth preserving.
- (2) Additional functions can be used for deriving new values that influence the truth of predicates.

- (3) The attribute-values can be used to influence case-based reasoning like nearest neighbour computation.

The last two methods are not always truth preserving.

22.4 Information Theory and Entropy

Often one is confronted with incomplete information. For instance, the truth value of a statement or the value of an attribute may be missing. In order to obtain such information an action has to take place. Often, such an action is costly, as for instance, with tests in medicine. Such situations occur regularly in conversational CBR, for example, where a new part of the query may be explored by asking the user for a value of a different attribute in the query. However, not all attributes are of the same quality in this respect. Some attributes may contain little new knowledge while others can contain something that is very close to the solution.

Information theory quantifies the amount of information contained in a knowledge package. This allows for looking for very informative attribute values. The problem is that this has to be decided a priori, when one has little information about the importance of the result. For instance, some values of the attribute may contain more information than others. This creates some uncertainty that is treated using probability theory.

We restrict ourselves to getting the value of an attribute and start with an example where we assume we have two Boolean attributes A and B such that

- $\text{Prob}(A = 0) = \text{Prob}(A = 1) = \frac{1}{2}$,
- $\text{Prob}(B = 0) = 0.9$, $\text{Prob}(B = 1) = 0.1$.

If we have the choice of getting the values of either A or B , we would take the value of A . The reason is that for B we almost know what will be happening. That means we should estimate the information provided by the value of A as higher than that obtained from B .

Now we come to the general theory. For defining the information provided by an event E , we need to define a certain measurable information $I(p)$.

Definition 22.8 Suppose E is an even number. If s is a possible outcome of E , the information content $\text{IC}(s)$ of this outcome is (with ld being the dual logarithm)

$$\text{IC}(s) = -ld(\text{Prob}(s)).$$

If there are n possible outcomes $O = \{c_1, \dots, c_n\}$ for an event E , the entropy $H(E)$ is the expected information content:

Definition 22.9

$$H(E) = - \sum_{i=1}^n \text{Prob}(c_i) \cdot ld(\text{Prob}(c_i)).$$

Definition 22.10 For two events A and B , the conditional information content of the event A after B has occurred is defined as:

$$IC(A|B) = -ld(P(B|B)).$$

Now the events describe the possible values of an attribute A . After choosing attribute A the situation is partitioned into sub-events according to values a of A . This is as in decision trees.

The remaining information I_{rem} is equal to the weighted sum of the amounts of information for the subsets:

$$I_{rem} = - \sum_a \text{Prob}(a) \sum_s \text{Prob}(s, I, a) ld(\text{Prob}(s, I, v)).$$

If we get the value of a new attribute A there is some new information $I_{new}(A)$. The remaining information is now $I_{rem} = I(P) - I_{new}$. The goal is now to arrange the attributes in such a way that ultimately these differences approach 0 as fast and cheaply as possible. The most “informative” attribute is the one that minimizes I_{rem} .

Now consider the case that attribute B completely solves the problem while asking A would result in further queries. The point is that one is looking at the problem to be solved. Returning to the example of the Boolean-valued attributes, we assume that the answer to predicate B gives the final answer to the problem. In this situation we would call B despite the fact that the entropy gives another recommendation. This leads to a refinement of the entropy concept.

Definition 22.11 If some earlier event B had the outcome b , then the conditional entropy for the event A with the outcomes a_1, \dots, a_n under the assumption of b is:

$$H(A|b) = \sum_{j=1}^m P(a_j) \cdot H(A|b).$$

Definition 22.12 If event B has the m disjoint outcomes B_1, \dots, B_m , then the conditional entropy of event A under the assumption B is:

$$H(A|B) = \sum_{j=1}^m P(b_j) \cdot H(A|b_j).$$

Replacing entropies by conditional entropies gives a concept for selecting the next attribute in a dialogue when the query is constructed incrementally.

The consequence for CBR is that this gives a guideline for the ordering of requesting values of attributes if one is interested in keeping a conversation as short as possible.

22.5 Utilities

The correctness and amount of information do not tell us if the obtained information is of interest to us or in a certain context. The utility has been a motivating concept throughout the book.

The theoretical foundation is utility theory. It has its origin and background in mathematics and economics, in particular, in the von Neumann-Morgenstern theory.

Essentially, utility theory considers situations where decisions have to be made and where the effect of the decision is uncertain. This means that the effect is known with some probability only. The more general view now is that the utilities are assigned to the effects rather than to the decisions.

Definition 22.13 The expected utility of a decision d is

$$u(d) = \text{Prob}(\text{eff}) \cdot u(\text{eff}).$$

Here eff is the range of the possible effects of d and $u(\text{eff})$ is its utility.

Utility has a relational and a functional form (also called ordinal and cardinal form), as introduced earlier. The relational form was introduced as a preference relation. The relational and the functional forms are strongly connected with each other. The simple explanation is that utility functions induce preferences.

The induced preference of a utility function is: b is preferred over c :

$$c \Leftrightarrow u(b) > u(c).$$

The other direction is a little more involved. For dealing with expected utility, the preference relation has to satisfy some axioms. For describing them we need some notation on partial orderings that is also used for optimization with respect to \geq .

We consider a set U and a binary relation R on U , i.e., a subset of pairs of elements of U . For $(x, y) \in R$, we write $R(x, y)$ or $x \geq y$.

Notation of basic partial ordering properties:

- \geq is reflexive (irreflexive) if always (never) $(x \geq x)$ holds.
- \geq is symmetric if $(x \geq y)$ implies $(y \geq x)$.
- \geq is transitive if $(x \geq y)$ and $(y \geq z)$ imply $(x \geq z)$.
- A partial order is a reflexive and transitive relation.
- \geq is asymmetric if $(x \geq y)$ implies $\neg(y \geq x)$.
- \geq is antisymmetric if $(x \geq y)$ and $(y \geq x)$ imply $x = y$.
- \geq is total (or complete) if for all $x, y, x \neq y, (x \geq y)$ or $(y \geq x)$ holds.
- \geq is a linear order for all $x, y, x \neq y$, exactly one of $(x \geq y)$ or $(y \geq x)$ holds.

A set of events $\{(A_1, p_1), (A_2, p_2), \dots, (A_n, p_n)\}$, with probabilities p_i is called a lottery. Now we can formulate the von Neumann-Morgenstern axioms for lotteries in terms of preferences \geq :

- (1) \geq is complete.
- (2) \geq is transitive.

- (3) Independence: Let A , B , and C be three lotteries with $A \geq B$ and let $q < p$; then $((A, p), (B, (1 - p))) \geq ((A, q), (B, (1 - q)))$.
- (4) (Continuity) For all A, B, C with $A \geq B \geq C$ there exists exactly one probability p such that $((B, p), (C, 1 - p))$ is as good as $(B, 1)$.

Continuity assumes that when there are three lotteries (A , B and C) and the individual prefers A to B and B to C , then there should be a possible combination of A and C in which the individual is then indifferent about this mix and the lottery B . If all these axioms are satisfied, then the individual is said to be rational. In this case one can show that the preferences can be represented by a utility function. This result is called the “von Neumann-Morgenstern utility representation theorem”.

When applying utility, one needs to know the involved utilities and probabilities. It is well known that utilities, even if they are precisely defined in terms of mathematical functions, are subjective. That means there is no universal model for utility. This means it depends on the user who is confronted with some context. A common error is to identify utilities and costs. If costs increase then the utility does not necessarily go up and the relation between the two may be nonlinear. Suppose, for instance, a company cannot survive if the costs are $> \$50,000$. Then utility function will be a step function.

In addition, the involved probabilities may not be based on a frequency model but again be of subjective character. That means the probabilities are based on the opinions and needs of users. That makes it hard to deal with them but opens a door for applying CBR again. See more in Chap. 16, Probabilities.

22.5.1 Optimization

The goal is always to maximize utility. Partial orderings are essential for any kind of approximate reasoning. The ordering describes the term better in an approximation formally.

For describing this we extend the notions of partial ordering:

Two elements x and y are

- Comparable if $(x \geq y)$ or $(y \geq x)$ holds.
- Indifferent if $(x \geq y)$ and $(y \geq x)$ hold.
- Equivalent if for all z :
 - $(x \geq z) \Leftrightarrow (y \geq z)$ and $(z \geq x) \Leftrightarrow (z \geq y)$ hold.
- An element z is a global maximum (or global minimum) if for all x $(z \geq x)$ or $(x \geq z)$ holds.
- For subset $V \subseteq U$ we put $\sup(V) = z$ if $(z \geq x)$ for all $x \in V$; and if $(y \geq x)$ for all $x \in V$, then $(y \geq z)$.
- $\inf(V)$ is defined in the same way.

Suppose (V, \leq) is a partially ordered set, and $X \subseteq V$ is a subset with $a = \sup(X)$.

For $x, y \in V$, $x, y \leq a$, the element y is called a better approximation than x to a if $x \leq y \leq a$.

Optimization is a central part of approximate reasoning. An optimization problem is of the form

$$\text{Maximize } f(x) \text{ under the constraints } C(x),$$

where f is a function with values in a partially ordered set (V, \leq) ; f is called the objective function. The constraints can be formulated in any way, not only in predicate logic. It determines the area in which the solution has to be located.

In practice, one often formulates problems in an imprecise, vague or incomplete way. This can be due to different reasons: A person may not be willing or able to formulate the problem precisely. The latter is often the consequence of the fact that one has to fill out a questionnaire that allows only restricted formulations. In addition, if there are several persons with different criteria, not all of them can be satisfied, and there will be complaints.

This is an area where one can apply experiences and CBR. In particular, it is a possibility for conversational CBR, where a similarity comparison between the user's original demands and the result obtained by the optimization procedure can take place.

22.6 Chapter Summary

The chapter contains mixed formalisms on topics discussed earlier in the book. The first type of formalism deals with knowledge representation. As a prominent example of knowledge representation, classical logic was discussed. We introduced syntax, semantics and reasoning methods in an axiomatic way. As a special case we also discussed the relation to attribute-value representations.

In order to deal with incomplete representations, we introduced information theory for guiding the search for missing knowledge. Besides that, the aspect of utility was formally introduced for dealing with optimality questions. For this, a basic terminology about partial orderings was presented.

22.7 Background Information

There are many textbooks on logic-oriented knowledge representation. As examples we mention Russell and Norvig (1995) and Genesereth and Nilsson (1987).

Entropy in this context was first used for building decision trees; see Quinlan (1993). The standard reference for mathematical utility theory is von Neumann and Morgenstern (1944).

Subjective probabilities and utilities have to satisfy some rationality conditions that can be formulated in terms of axioms. An influential paper was Savage (1954). Fishburn (1986) provides a fairly complete picture of the present state of the measurement-theoretic foundations of subjective probability.

References

- Fishburn PC (1986) The axioms of subjective probability. *Stat Sci* 1(3): 335–345
- Genesereth MR, Nilsson NJ (1987) Logical foundations of artificial intelligence. Morgan Kaufmann, San Mateo
- Quinlan JR (1993) C4.5: programs for machine learning. Morgan Kaufmann, San Mateo
- Russell SJ, Norvig P (1995) Artificial intelligence: a modern approach. Prentice Hall, Upper Saddle River
- Savage JL (1954) Foundations of statistics. Wiley, New York
- von Neumann J, Morgenstern O (1944) Theory of games and economic behavior. Princeton University Press, Princeton

Chapter 23

Relations and Comparisons with Other Techniques

23.1 About This Chapter

This chapter is addressed to readers who are interested in the relation of CBR to other areas. CBR systems have more or less close connections to other methods. It depends on the problem and the interests of the reader on how to make a choice between the methods. We discuss areas that have influenced CBR significantly, that are in competition with CBR or that are candidates for cooperation. This is a background chapter and it is assumed that the readers have some knowledge about the discussed topics and the previous chapters. As previously mentioned, this chapter does not have a tools section.

23.2 General Aspects

In this chapter, we sketch other approaches insofar as they have a relation to CBR. The intention of this is twofold:

- Finding out under which circumstances one should choose which methodology or which ones can be combined.
- Identifying origins and influence factors of CBR.

We distinguish between methods that could be competitors or collaborators of CBR and areas that have influenced CBR. The former are database management systems, information retrieval systems, pattern recognition systems, knowledge-based systems, and machine-learning systems. The latter are uncertainty, cognitive science, knowledge management, and, again, machine learning. Figure 23.1 gives an overview.

We investigate the techniques step by step, where some attention is paid to knowledge containers that exist in the other techniques too. There are structural commonalities and differences as well as different advice for when they should be used.

In Fig. 23.1 we put CBR in the centre from the viewpoint of this book. There is no sharp boundary with other techniques; often, it depends on the specific application.

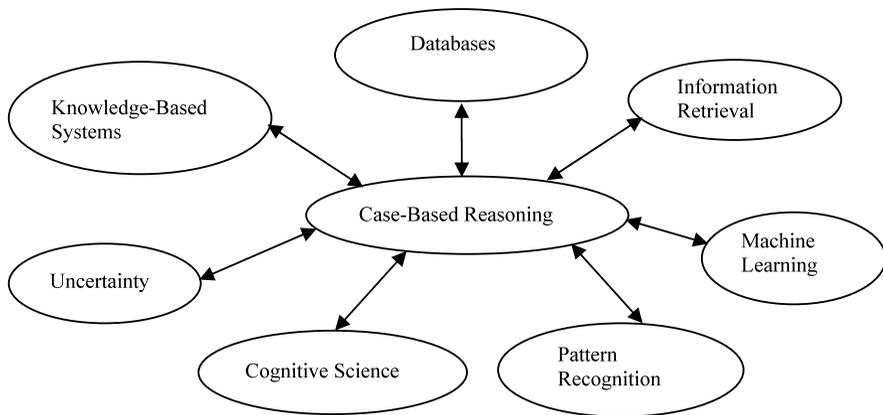


Fig. 23.1 Relations of CBR to other areas

Some approaches have advantages for certain tasks and disadvantages for others. Next, we will discuss some of the indicated relations. There we distinguish between types of systems:

- Systems with retrieval engines.
- Knowledge-based systems without special retrieval engines.

23.3 Systems with Retrieval Engines

In this category we consider methods where certain solutions are retrieved from a base. These are database systems, information retrieval systems, and pattern recognition systems.

23.3.1 Database Management Systems

There is an obvious connection between database management systems and CBR systems. Both contain data elements of interest that one wants to retrieve if needed.

Database queries pull out only items that exactly match certain retrieval criteria mentioned in the query. On the other hand, CBR queries pull out items that are similar to our criteria, even if they do not match exactly. CBR queries rank or grade items retrieved using similarity.

Another major difference is that one does not consider reuse and adaptation in databases.

Regular database queries return historic facts unaltered while CBR queries can return a proposed solution that has been adapted from past solutions to fit the current

Table 23.1 Databases versus CBR systems

Database systems	CBR systems
Simple search: All or nothing	Search for most similar objects
Underspecification, too many answers	Number of retrievals can be specified
Overspecification, no answers possible	
No specific domain knowledge needed	Domain knowledge in the containers
Efficient and correct answers	Search can be slow, correctness not ensured
Closed world	Open world
Static when in use	Can improve incrementally
Long transactions allowed	In principle no long transactions intended
No conversations	Systems can have conversations
No adaptation	Adaptation intended

situation. As a consequence, database technology is concerned with closed worlds while CBR can deal with open worlds.

In addition, many CBR systems offer a conversational approach to formulating a query while in databases the query has to be provided in one step. All these facts refer to the fact that databases have a black box character.

Both databases and CBR systems can be improved over time. However, CBR systems pay more attention to this because improvement is part of the process model.

In database systems the knowledge containers are the stored data and the retrieval functions. Traditionally, the latter have been concerned with efficiency and safety. Today, there is also a semantic problem because of heterogeneous databases. These databases do not, for instance, have consistent uniform terminology and use different levels of abstraction.

Databases have wider application domains and are used much more often in everyday life than are CBR systems. The main reason is that databases are absolutely necessary today while CBR systems are still “only useful” for more sophisticated tasks. Because the changing environment requires immediate reactions, long transactions have to be performed by databases that need specific methods not present in CBR systems.

Often, databases and CBR systems are combined. A major reason is that in application scenarios cases were stored in a database before one was thinking of using CBR. As a consequence, one needs a connection between the database and the CBR system that allows fast transfer. In Table 23.1 one finds a short summary.

23.3.2 Information Retrieval Systems

Information retrieval (IR) systems have in some sense the same goal as CBR systems because they are both trying to retrieve the best possible answers to queries. There is an even closer relation to textual CBR because both refer to written text.

Information retrieval systems are in some sense more restricted than CBR because they are retrieving documents only. Therefore, one is searching for documents. This is done with the hope that the documents contain information for solving a certain problem. Therefore, this could be better called document retrieval. However, because one is searching a document for a specific reason, the term is not quite wrong. In fact, in recent years the methods are developing as knowledge retrieval. Content can be considered only if certain terms of interest are extracted a priori and added to the document description.

The document collections are often quite large, and the documents are not collected for a specific purpose. A typical example is a search on the Web, where most documents are not of interest to a user. This makes retrieval difficult and the retrieval engines are the most important part of an information retrieval system.

In contrast to databases, one does not have to name a specific document. Instead, one can present an arbitrary term of interest. If this term occurs in the document title or description then the document is a retrieval candidate.

IR systems have to model their document bases in order to allow retrieval. This is an index structure, as discussed for CBR systems. A very popular model is the vector space model that has been widely used in the traditional IR field (also discussed in Chap. 17, Textual CBR. It is comparable to flat attribute-value CBR systems. The vector contains terms that are weighted. It tries to make “intelligent” search in unstructured document sets possible, especially on an intranet. For this, content-oriented measures have been developed. The close relation to textual CBR is illustrated by the fact that both can make use of WordNet; see Chap. 17, Textual CBR.

Although the vector space approach looks similar to the attribute-value representation in CBR, there are some differences:

- (i) In CBR, we have attributes that have a domain from which the values are taken.
- (ii) In IR, the vector coordinates are just numbers; there are no variables that could be instantiated and each coordinate is labelled by a fixed term.
- (iii) The information retrieval (IR) view is strongly document-oriented. In addition, the behaviour of users is studied and used in the retrieval. Not much knowledge is used in IR; it is mostly of statistical character.
- (iv) In fact, all major IR models heavily make use of statistics, counting of index terms, and so on:
 1. The index vocabulary is determined based on the given document collection by applying some statistics and heuristics. For example, terms that occur too often are considered to be useless, as are terms that occur too seldom.
 2. The weight of index terms is again based on counting frequencies in a given document collection. Consequently, the similarity of documents is based on these frequencies, too. If the term does not occur in the document, its value is 0.

The knowledge in IR systems is defined in the form of weights of the terms. These weights come either from a statistical analysis or are provided by the user with the hope that interesting documents are retrieved (which is rarely done).

The weights defined on a statistical basis are, for instance:

- Term frequency, f_{ij} , the number of occurrences of term y_j in document x_i .
- Inverse document frequency, $g_j = \log(N/d_j)$. N is the total number of documents in the collection and d_j is the number of documents containing term y_j .

Mostly, IR systems can be recommended on the general context level, as introduced in Chap. 13, Advanced Similarity Topics. There, one has many documents and little structure. The CBR alternative is more restricted to the lower context levels, where more specific knowledge is available that the system can make use of.

The knowledge containers of information retrieval systems are the document base, the similarity measure and the retrieval engine.

The retrieval result is, however, not necessarily convincing. Therefore, a process starts for improving the results. The technique used is like the one for query adaptation discussed in Chap. 9, Adaptation. That means the query is reformulated into an optimal query that (hopefully) finds all the relevant documents.

For the reformulation, there are several ways to do it. In principle, always some new words are added to the query. The techniques are categorized based on how words are added to a query. They can be obtained from external sources like a thesaurus, controlled vocabulary, or ontology, or from the texts being searched based on a statistical relation to a query. Such methods have been extended to textual CBR, where they are used to enhance the vocabulary. There is also a relation to conversational CBR.

A popular method for improving IR systems is query expansion, where terms are added to the query that are similar to the given ones. Automatic query expansion uses a frequency measure on pairs of words, i.e., some kind of co-occurrence measure.

Like other methods used in textual CBR that rely on ideas designed for IR, enhancing the vocabulary container is no exception. Ideally, for literary texts, we would like to add terms such as “resembles”, “is similar to”, and “is analogous to”, to enhance understanding of “bears a faint resemblance to”. Unfortunately, in practice it is not that easy to find those semantically similar expressions in an automated fashion.

In summary, one can say that from the CBR process model only problem formulation and retrieval can be found in IR. There is, in particular, no reuse and adaptation. In Table 23.2 a summary is shown.

23.3.3 Pattern Recognition Systems

Pattern recognition (PR) can be viewed as a competitor to CBR in certain areas. The data orientation is important for pattern recognition. These data can be, however, of a very general nature, for example, numeric or symbolic.

Table 23.2 CBR systems versus IR systems

Information retrieval engines	CBR systems
Indexing of internet	Specific for a context
Can search on the Web	Search restricted to parts of the Web
Indexing by keywords	Indexing by features
No specific domain knowledge needed	Domain knowledge necessary
Heavily based on statistical knowledge	User domain and background knowledge
Query expansion	Conversational CBR
No adaptation	Adaptation intended

The major differences with CBR are:

- (a) The emphasis is mainly on the retrieval step in the CBR process model.
- (b) PR uses black box models.
- (c) The applications are usually very specialised; their main task is to perform a match.
- (d) Symbolic reasoning methods do not play such a role.
- (e) The measures are usually very simple and allow fast computation.
- (f) It does not have an explanatory character.
- (g) Statistics plays a major role.

On the one hand, this restricts the scope of possible applications. On the other hand, very successful and extremely efficient methods have been developed. In fact, in many applications pattern recognition methods are unbeatable for this reason. Popular applications are, for instance, fingerprint or face identification, and speaker identification. It is not intended for CBR to compete with pattern recognition in such areas. We comment on some methods that are of interest to CBR too and on some commonalities and differences with CBR.

The area includes subdisciplines like discriminant analysis, feature extraction, error estimation, cluster analysis (sometimes called statistical pattern recognition), grammatical inference and parsing (sometimes called syntactical pattern recognition).

A typical pattern recognition system makes its “decisions” by simply looking at one or more feature vectors *feat* as input. Pattern recognition aims to classify data (patterns) based either on a priori knowledge or on statistical information extracted from the patterns. The patterns to be classified are usually groups of measurements or observations, defining points in an appropriate multidimensional space. The strength of this approach is that it can leverage a wide range of mathematical tools ranging from statistics to geometry to optimization techniques. There are two tasks:

- (a) The representation task: How to represent the objects in a suitable way?
- (b) The recognition task: How to classify the represented objects?

PR and CBR are closely related when images have to be retrieved. In PR, images are represented by certain geometric objects that are characteristic of the images un-

Table 23.3 PR systems versus CBR systems

Pattern recognition systems	CBR systems
Simple representation languages	Representation languages can be complex
Simple similarity measures	Similarity measures can be complex
Indexing by keywords	Indexing by features
Knowledge in the feature definition and the probability definition	Domain knowledge in all knowledge containers
Knowledge in the feature definition and the probability definition	Domain knowledge in all knowledge containers
Heavily based on statistical knowledge	User domain and background knowledge
Query put only once	Conversational CBR possible
No adaptation	Adaptation possible

der consideration. An example is distances between the eyes for faces. Such objects are extracted and represent the image.

An important point is that the representation phase is a closed phase; after it is finished it cannot be taken up again. When this step is performed, there is a crucial assumption: All remaining knowledge is contained in the data of the examples, i.e., no background knowledge can be used afterwards.

The knowledge in the data is concerned with:

- (1) The dependencies between the attributes: Here, often Bayesian networks are employed.
- (2) The determination of the decision surfaces of the classes.

For the recognition task, statistical techniques have been widely used; some occur in CBR systems too. Statistical classifiers include linear discriminant functions (LDFs), quadratic discriminant functions (QDFs), Parzen window classifiers, nearest neighbour (1-NN) and k -NN rules, and so on.

The high dimensions of attribute vectors is a topic that deserves special attention. The reduction to lower dimensions requires the removal of irrelevant data. For more on this, see Chap. 18, Images.

As a consequence of using the simple similarity measures, the decision surfaces are quite difficult; they are, in particular, far from being linear. We remind the reader that in CBR the introduction of virtual attributes shifts the nonlinearity of the measure to the attributes. If the surfaces are under control then there is no need to invent virtual attributes or sophisticated similarity measures. Everything is shifted to identify the decision surface. Table 23.3 shows a comparison.

23.3.4 Knowledge Comparison for Retrieval Systems

This comparison is concerned with the knowledge used in the systems and where it is contained. Table 23.4 shows a comparison between CBR, IR and PR.

Table 23.4 Different retrieval systems

Methodology	Background knowledge	Knowledge in the data
Case-based reasoning	In all knowledge containers	Only for improving measures
Pattern recognition	For the representation task	In the recognition phase
Information retrieval	In (the frequency) of terms	In the weights and the query expansion

23.4 Explicit and Implicit Knowledge Representation

In this category we consider systems that represent knowledge or are intended to achieve such a representation. They have special retrieval systems incorporated. The knowledge units one wants to retrieve are quite often consequences of the knowledge that is stored. For that reason deductive methods and learning play an essential role.

As a first type we have knowledge-based systems and knowledge management. The second type we consider are machine learning systems.

23.4.1 Knowledge-Based Systems

Knowledge-based systems play a double role in relation to CBR systems. On the one hand, they provide influence factors for CBR systems. On the other hand, they are competitors and collaborators with CBR.

Knowledge-based systems make an important distinction between explicitly and implicitly stored knowledge. In contrast to CBR, their most important task is retrieving the implicitly stored knowledge. For this there is no specific search machine involved. Instead, a logical inference process is started and that is the kind of reasoning that takes place. It has the advantage that the inference process guarantees the validity of the inferred results (as long as the inference methods are correct). In addition, the inferences can be very long and complex. In this respect these systems are superior to CBR.

A knowledge-based system is in principle a logical system and therefore the representation formalisms are essential. For instance, in a rule-based system like the ones developed using Prolog, we have the containers facts and rules. The rules allow knowledge to be inferred from the facts. The knowledge shifting can take place from facts to rules: One starts with many facts and few rules and ends up with fewer facts and more rules.

Knowledge bases can be used as a black box and can be built into larger processes without human interaction. Hence, if one wants very reliable solutions then rule-based systems have an advantage. In dynamically changing contexts this is, however, somewhat different.

The logic orientation has not only advantages but also drawbacks:

- (a) Firstly, the system has to be fully understood before it can be used. The knowledge is compiled.

Table 23.5 Knowledge-based systems versus CBR systems

Knowledge-based systems	CBR systems
Experts are replaced	Users are supported
New knowledge by inference	New knowledge by adaptation
Knowledge implicitly stored	Knowledge explicitly stored in cases and implicitly in containers as similarity measures
When building the system expert knowledge is needed	For building the case base no expert knowledge needed
Updating problematic	Maintenance easier and more systematic
Aimed at 100 % domain knowledge	Works with partial knowledge
Gives valid answers	Can give approximate answers

- (b) The knowledge acquisition process is often quite involved and time consuming.
- (c) Due to its black box character, one does not and cannot have an insight into its internal structure, that is, into the problem-solving process. This makes updating difficult.
- (d) It is difficult to deal with utility.
- (e) It is difficult to dynamically change contexts.

On the other hand CBR systems:

- (f) Require less general knowledge because the knowledge in the cases does not have to be fully understood.
- (g) The knowledge in the other three containers has to be acquired. This can be done, however, at a later time. One can start with insufficiently filled containers and still obtain a working system.
- (h) The similarity orientation allows working in degrees.
- (i) CBR can deal with approximation, where knowledge-based systems have problems. On the other hand, CBR can make use of techniques of the two types of systems.

We show a comparison in Table 23.5.

23.4.2 Machine Learning

In machine learning one considers data as examples. It was pointed out several times in this book that one does this in CBR too. These examples contain implicit knowledge and learning should make it explicit. In contrast to logical deduction, not all inferred statements may be valid, as in CBR.

Machine learning systems and CBR systems have several aspects in common:

- They operate on presented examples that we call experiences.
- Both aim also at avoiding or reducing knowledge acquisition, which is a standard bottleneck in knowledge-based systems.

Table 23.6 Machine learning versus CBR

Machine learning systems	CBR systems
General rules and laws are generated	Specific solutions are generated
Results are not precise or certain	Results are not precise or certain
Degree of falsehood cannot be grasped	Similarity controls falsehood
Unsupervised learning possible	Unsupervised problem solving cannot be done

- Both do not guarantee the correctness of a solution.
- Both aim also at discovering properties:
 - Machine learning systems discover generalities of examples;
 - CBR systems discover knowledge for similarity by looking at other examples.

Machine learning is an eager approach. This is due to the fact that machine learning systems compile knowledge from all examples into a general form from which properties of individual problems can be instantiated. This means machine learning learns concepts that are applied later on.

In contrast to this, CBR is partially a lazy approach. This refers to the case base container, where the knowledge in the cases is only accessed when it is needed by a user.

An advantage of building CBR systems is that many aspects do not have to be initially understood. However, the price to pay is that the system sometimes is initially not very good. One role of machine learning is to improve the performance of the CBR system. CBR systems and machine learning techniques have mainly two relations to each other:

- They can benefit in different ways from each other.
- They can be used as alternative techniques for problem solving.

The benefit is, for instance, that the special form of the container contents can be learned. This is detailed in Chap. 10, Evaluation, Revision, and Learning. Another difference to CBR is that learning methods can be unsupervised what is not directly possible for CBR. A consequence is that CBR can be combined with unsupervised learning, as, for instance, clustering. Table 23.6 shows a comparison.

Finally we compare how knowledge is discovered in the different approaches. The comparison with respect to getting results is shown in Fig. 23.2.

23.5 Influence Factors

In some way, the broadness of CBR results from its various influence factors. This was a process over decades that is still going on. Despite the many changes in details of the methods, the basic elements are still living. These main elements are the process model and the knowledge containers.

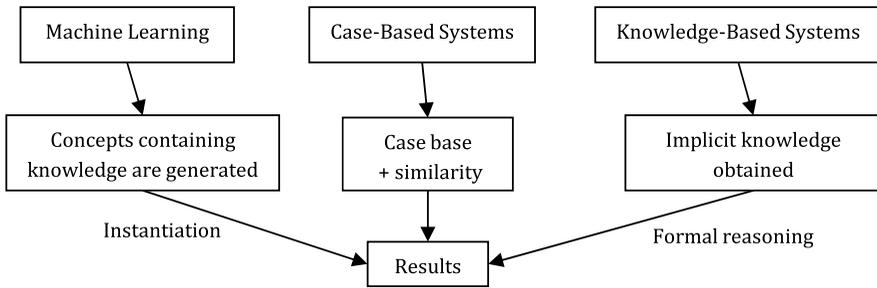


Fig. 23.2 Comparison of methods

In this category we look at cognitive science, knowledge-based systems, analogical reasoning, and uncertainty systems. All of them have had a significant influence on the development of CBR.

23.5.1 Cognitive Science

This section extends the corresponding more historical Sect. 1.2.1, in Chap. 1. In the beginning, the aspects of cognitive science were human-oriented. It was always clear that intelligent human reasoning is not restricted to and definitely not coincident with correct logical reasoning. Later on, the interest included reasoning with computer support. This had an essential influence on development of CBR as a whole. The inherited views from cognitive science have been merged with computational techniques.

Some of the basic roots of CBR that are located in cognitive science have a relation to cognitive psychology. They deal with solving complex problems by looking at human behaviour. For this reason one was creating special cognitive models that were concerned with the way humans understood and memorized experiences and analysed them for further use. These models have been implemented in various cognitive architectures.

One of the roots is the MOPs model of dynamic memory, which is based on the central concept MOPs. MOPs originated from the concept of scripts. A script is a memory structure that stores general knowledge about typical situation patterns. A script, and later MOPs, represent classes of situations, and include the events that occur in those situations, their typical goals, and the participating actors. The intuition is that any agent can follow it to act accordingly. What distinguishes scripts from MOPs is that MOPs are dynamic memory structures (i.e., they evolve) interconnected in networks. Cases are specific instances of situations that occur in the context of a MOP.

In this memory structure the special situations are represented as cases. The structures are concerned with storing general background knowledge as well as special situations because for solving a problem one needs a combination of both. There are two kinds of this memory structure:

- Memory Organisation Packets (MOPs).
- Thematic Organisation Packets (TOPs).

MOPs deal with dynamically changing situations. The term dynamic memory comes from the idea that understanding an experience yields a reorganisation of the memory structure. MOPs deal with sequences of past scenes, each specially stored. TOPs contain general knowledge of the relations between goals and event sequences across contexts.

The basic assumptions in dynamic memory are:

- Memorizing and adaptation are essential processes for understanding.
- Indexing is central for memorizing.
- Understanding leads to reorganising the memory and is a crucial dynamic process.
- The memory structures for knowledge processes and knowledge storage are the same.

Special MOPs are E-MOPs, episodic MOPs. They are concerned with storing and indexing episodes; the memory can contain episodes as experiences that are cases. Experiences can also change the dynamic memory by storing new cases, changing index structures, generating new MOPs and TOPs, and using cases. These developments have slowly merged into CBR.

23.5.2 Analogical Reasoning

Analogy was discussed in several parts of this book. Some technical details can be found in Chap. 9, Adaptation, in the context of reusing solutions. Analogical reasoning has a long history. It has many overlaps with CBR and has had influence on the development of CBR. Both have in common the way in which they make use of past experiences and there is no clear borderline between the two. Essential differences are:

- CBR relies on the concept of similarity, which focuses on the attributes shared by two entities. Analogy is based on comparing the abstract structure of two entities.
- Analogical reasoning typically compares entities from different domains; CBR case bases are usually within one same domain.
- It suffices for CBR that one similar case can be used to solve a new problem; for this purpose, a specific example is generally all that is needed. Analogy examines principles or concepts of two situations aiming at learning and generalization. This takes place on a general level and is not aimed at solving individual problems.
- Analogy operates on a more strategic level than CBR, where it can give insights to reasoning by providing cognitive models of analogy. In contrast to that, CBR is mostly aimed at developing powerful computer systems for reasoning and problem solving.

Table 23.7 Analogy versus CBR

	Analogy	CBR
Goal	Modeling of analogy for human reasoning	Generation of efficient computer systems
Domains	Across domains	Inside a domain
Similarity	Comparison of abstract structures	Specific similarities
Memory	General	Cases
Retrieval	On a semantic basis	Indexing
Orientation	Cognitive science	Several

- Operationally, analogical reasoning provides a unified view; in contrast, CBR breaks the solution process down into steps.

The term analogy is frequently used in mathematics. But this was never developed as a formulated reasoning method and was always addressed to human thinking.

Analogical mapping is the theory for crossing domains. This supports again the unified view of analogy. Table 23.7 shows summary sketch of the comparisons.

23.5.3 Uncertainty

Uncertainty is a general area with very different types of systems. The basic relation to CBR is that CBR has an intrinsic way to deal with uncertainty because of the similarity concept. Technically, a similarity measure is simply a fuzzy set on ordered pairs. On the one hand, CBR can solve certain problems in uncertainty areas. The similarity assessment as well as solutions and evaluations make use of uncertainty techniques. There are several aspects that fuzzy sets and CBR systems have in common; they are discussed in Chap. 15, Uncertainty. Here, we just mention that both measure certain properties in terms of degrees: Degrees of membership and degrees of similarity.

Probability is a specific kind of uncertainty. Its basic view is to look at frequencies. That has led to mathematical models and statistics. Concepts of statistics give rise to similarity measures, as discussed in Chap. 16, Probabilities.

Similarity measures, fuzzy degrees, and probabilities have in common that they are based not only on models but on subjective opinions too.

One way to compare them is by looking at the efficiency and at the problems to which they are applied. In CBR, usefulness is central and much attention has been put on it. In contrast to this, fuzzy degrees and probabilities are not directly connected with usefulness. They merely record observations and opinions.

Another approach is to compare the way in which one has belief in the results. Of course, if the input to a system is garbage, the output will be too. Therefore, in a comparison we have to assume that the input is correct or the error can be estimated. This is discussed in Sect. 12.5.4 on provenance.

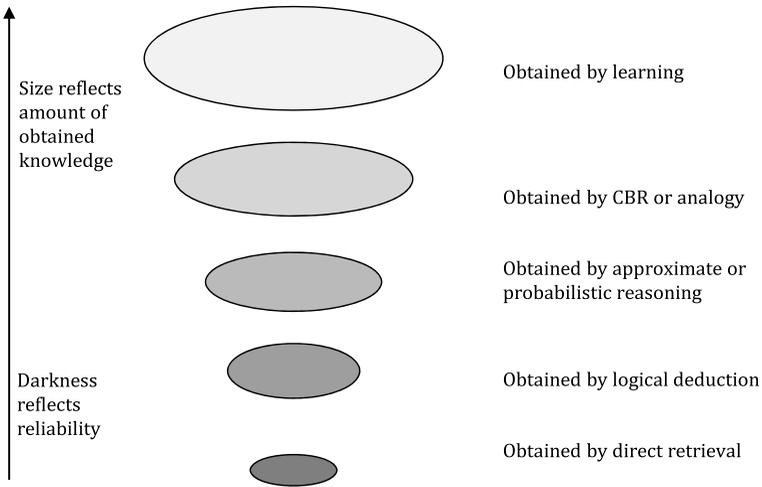


Fig. 23.3 Number of results versus reliability

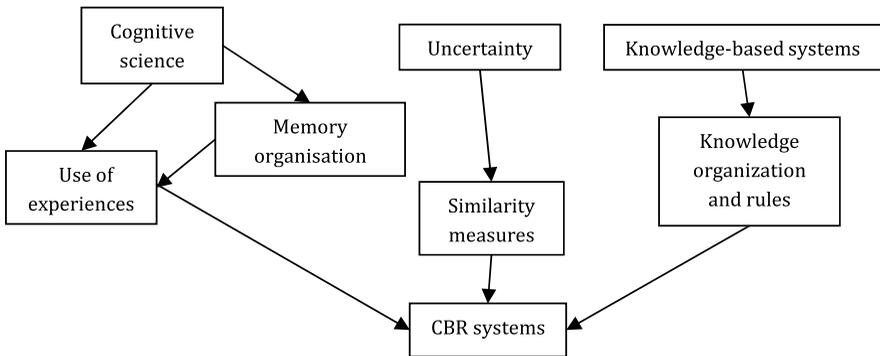


Fig. 23.4 Multiple influences of other disciplines

The general principle is that certainty depends on the certainty of the assumption and the reasoning. A consequence is that the less we know, the less certain we are that the consequences are true. On the other hand, the more we restrict ourselves to true consequences, the less useful may be the results we get. Figure 23.3 illustrates this. When looking at this diagram, one has to realise that high reliability cannot always be obtained. In such situations, one has to be satisfied with less reliable methods that can be very useful. There is a parallelism between getting more results and the increase in the level of uncertainty. In addition, uncertain information may be better than no information at all.

We illustrate the different kinds of influences of cognitive science, uncertainty, and knowledge-based systems on CBR in Fig. 23.4.

23.6 Chapter Summary

In this comparison we distinguished two questions. The first one considers areas that have systems which can be regarded as competitors or alternatives to CBR systems. The other one considers areas that are conceptually close to CBR and have influenced CBR with their concepts.

We looked at the following categories:

- Systems that retrieve results.
- Systems aimed at explicit knowledge representation and inference.

Possible competitors are database management systems, information retrieval systems, pattern recognition systems, and knowledge-based systems. Machine learning systems play more of an intermediate role; they and CBR can support each other. In particular, machine learning can improve all CBR knowledge containers.

As roots and ancestors of CBR, we considered cognitive science, analogy, knowledge-based systems, and uncertainty.

23.7 Background Information

This chapter is of background character. Therefore, we restrict ourselves to remarks about additional literature.

A detailed analysis of the relations of CBR to uncertainty is given in Hüllermeier (2007), where the relation of CBR to fuzzy sets is discussed in detail.

An early basis for CBR is the work of Schank, introducing scripts (Schank and Abelson 1977). Dynamic memory theory and memory organisation packets are presented in Schank (1982). A basic reference for cognitive views on CBR extending the notion of scripts and dynamic memory is in Kolodner (1993).

The connections between the cognitive aspects of CBR and the later more application-oriented views were explored in Dubitzky et al. (1996). The script theory is a special form of the schema theory in modern psychology.

The close relation between CBR and analogy is discussed in Burstein (1989) and Weber (1994). For the relations between machine learning and CBR, we refer you to the discussion in Chap. 10, Evaluation, Revision, and Learning; see also Li (1993).

References

- Burstein M (1989) Analogy versus CBR: the purpose of mapping. In: Hammond K (ed) Proceedings of a workshop on case-based reasoning, Pensacola Beach, FL, 1989. Morgan Kaufmann, San Mateo, p 133
- Dubitzky W, Hughes JG, Bell DA (1996) Case memory and the behaviouristic model of concepts and cognition. In: Smith I, Faltings B (eds) EWCBR-96: advances in case-based reasoning. Third European workshop, Lausanne, Switzerland, November 1996. Lecture notes in computer science (lecture notes in artificial intelligence), vol 1168. Springer, Berlin, p 120

- Hüllermeier E (2007) Case-based approximate reasoning. Springer, The Netherlands
- Kolodner JL (1993) Case-based reasoning. Morgan Kaufmann, San Mateo
- Li X (1993) Potential analysis for massively parallel computing and its application to neural networks. Dissertation, University of Kaiserslautern
- Schank RC (1982) Dynamic memory: a theory of reminding and learning in computers and people. Cambridge University Press, New York
- Schank RC, Abelson RP (1977) Scripts, plans, goals and understanding: an inquiry into human knowledge structures. Erlbaum, Hillsdale
- Weber G (1994) Fallbasiertes Lernen und Analogie. Beltz Psychologie Verlags Union, Weinheim

Index

A

Aamodt A, 15, 40, 245, 372, 441, 485, 503
Abasolo C, 164
Abdala DD, 441
Abelson RP, 538
Adaptation, 30, 31, 189–191
 lazy, 204
 numerical, 210
 planning, 202–209
 process, 199
 quality, 216
 search space, 206
 sequences, 200–202
 several cases, 210–213
 solution process, 213–215
 types, 198, 199
Adaptation container, 36, 106, 216, 238, 254,
 278, 316, 333, 402, 436, 458, 469
 maintenance, 265
Adlassnig K-P, 441
Aesthetics, 428–430
Aha DW, 15, 84, 245, 484, 485, 504, 505
Albert M, 245
Al-Emran A, 84
Aleven V, 84
Althoff K-D, 84, 186, 187, 272, 504
Analogical reasoning, 534
Analytic tasks, 53
Arcos JL, 164, 220
Ashley KD, 84, 407–409
Assignment, 511
Atayero AA, 463
Attribute, 27
 domain, 93
 relative relevance, 28
 type, 93
Auriol E, 186

Ayo CK, 463
Azeta AA, 463

B

Backward chaining, 205
Badra F, 245
Bag of words, 386
Balaa ZE, 441
Bang Y, 245
Barcia RM, 338, 408, 442
Bareiss R, 15, 338
Bayesian learning, 242
Bayesian network, 367
Bayesian reasoning, 365
Becerra-Fernandez I, 84, 297, 505
Bell DA, 537
Bello-Tomás JJ, 164
Benwell GL, 441
Bergmann R, 15, 52, 111, 164, 186, 272
Bertolotto M, 441
Bichindaritz I, 298
Birk A, 504
Boerner K, 164
Bogaerts S, 318
Bottrighi A, 463
Bradshaw S, 409
Branting LK, 407, 485
Breen S, 272
Breslow LA, 484, 485
Bridge attribute, 45
Bridge DG, 40, 84, 245, 273
Brüninghaus S, 407, 409
Bui Y, 504
Bunke H, 164
Burke R, 52
Burkhard H-D, 338, 408
Burststein M, 537

C

Call centres, 70, 466
 Campbell JA, 219
 Carbonell JG, 40, 219
 Carswell JD, 441
 Case, 18
 fuzzy, 348
 generalized, 107
 outcome, 21
 pivotal, 209
 problem, 20, 21
 solution, 20, 22
 Case acquisition, 251
 Case base, 18, 25
 conditions, 292
 distributed, 295
 Case base container, 36, 229, 253, 333, 395,
 433–436, 450, 469
 maintenance, 264
 Case-based reasoning, *see* CBR
 Case organisation, 25
 flat organisation, 25
 structured organisation, 26
 unstructured organisation, 26
 Case representation, 23, 24, 90
 attribute-value, 91
 Case retrieval nets, 321–325, 397
 CBR
 consists of, 17
 cycle, 5, 8, 32–34, 39, 42, 167, 189, 191,
 217, 257, 261, 425, 445, 447, 491, 492
 reasoning, 18
 steps, 4
 CBR maintenance, 260
 CBR systems
 correctness, 293
 properties, 288
 Chakraborti S, 338
 Chang L, 245
 Change dependencies, 270
 Change impact, 268
 Change management, 267
 Chatterjee N, 219
 Cheetham W, 355
 Chen H, 338
 Classification, 20, 55–57
 Closing operator, 428
 Clustering, 241
 Cognitive science, 3, 4, 523, 533
 Collaboratories, 498
 Combining CBR, 257
 Compatibility triangle, 303
 Completeness, 92
 Concordance, 361

Concordant, 303
 Conditional probability, 364
 Configuration, 20, 64–66
 Constraint, 514
 Content-oriented view, 376
 Context, 281–284
 levels, 283
 Conversational CBR, 460
 applications, 482
 commercial use, 482
 domains, 482
 goal, 467
 interaction, 467
 processing the initial description, 470
 Conversations, *see* Dialogue
 Copula, 361
 Correlation coefficients, 359
 Cosine, 401
 Covariance, 359
 Coverage, 206
 Covering theorem, 230
 Cox MT, 219
 Cummins LL, 273
 Cunningham P, 15, 219

D

Database management systems, 524
 Data-intensive, 281
 de la Rosa JL, 52
 Decision making, 488
 Decision trees, 10, 183–185, 258, 473, 476
 Deduction, 515
 Deepest common predecessor, 153, 399
 Degoulet P, 442
 Delany SJ, 84
 Derwand G, 187
 Design, 69, 70
 Development
 process steps, 255
 systematic, 254
 team, 255
 Di Baja GS, 441
 Diagnosis, 20, 55, 57–62, 459
 Dialogue, 465
 actions, 471
 architecture for, 476
 case bases, 481
 evaluation, 478
 formalisms, 472–474
 images in, 480, 481
 length, 475, 476
 management, 470–472
 quality, 478

- Díaz-Agudo B, 40, 164, 273, 298, 338, 408, 485
- Dimension reduction, 417
- Dissimilarity, 334
- Distributed representations, 387
- Diversity, 334, 335
- Do E, 441
- Document-oriented view, 376
- Domeshek E, 219
- dos Santos T, 441
- Dubitzky W, 537
- Dynamic memory, 5, 533
- Dynamic time warping, 161
- E**
- E-commerce, 42–46, 72–75
- Efficiency, 92
- Entropy, 516
 - conditional, 517
- Errors, 458
- Escalera S, 246
- Euclidean distance, 131
- Evaluation, 223, 251
- Evaluation, revision and learning triangle, 222
- Expected predictive power, 144
- Expected value, 308
- Experience factory, 248, 258, 259, 266, 269, 270, 295, 436, 493
- Experience mining, 6
- Experiences, 18
- Extended CBR view, 42, 50
 - comparison, 50
- Extended view, *see* Extended CBR view
- F**
- FAQs, 379
- Faupel B, 84
- Feature extraction, 392, 447, 454, 456
- Fish and shrink, 326
- Footprints, 333
- Forbus KD, 187
- Formalism
 - semantics, 510
 - syntax, 510
- Frucci M, 441
- Fuchs B, 219
- Functional dependency, 313
- Funk P, 463
- Fuzzy
 - defuzzification, 122, 343, 347
 - fuzzification, 343, 347
 - integral, 331
 - predicate, 331
 - Fuzzy classification, 351
 - Fuzzy set theory, 344–347
 - Fuzzy sets, 342–344
- G**
- Gardingen D, 298
- Genetic algorithms, 240
- Gentner D, 187
- Globig C, 245
- Göker M, 272
- Gómez-Gauchía H, 485
- González-Calero PA, 40, 164, 273, 485
- Gonzalvo X, 463
- Grammar, 385
- Graphs, 102
 - distance, 153
 - edit, 151
 - isomorphism, 149
 - prototype, 151
 - representations, 149, 422
 - similarities, 149
- Greene F, 15
- Gresse von Wangenheim C, 273, 504
- Griffiths AD, 245
- Grimnes M, 441
- Gross M, 441
- Gu M, 485
- H**
- Haddad M, 441
- Haigh KZ, 84
- Hamming measure, 126, 138
- Hammond K, 15, 219
- Han I, 246
- Hanney K, 219
- Henke AL, 187
- Heterogeneity, 295
- Hirst–St-Onge measure, 399
- Holographic reduced representations, 391
- Holt A, 441
- Hu D, 219
- Hughes JG, 537
- Hüllen J, 164, 219
- Hüllermeier E, 538
- Hwang I, 245
- Hybrid, *see* Combining CBR
- Hypertext, 379, 395
- I**
- IB1, 230, 232, 264
- IB2, 230, 232, 264
- IB3, 231, 232, 237, 264
- Iglezakis I, 273
- Ikhu-Omoregbe NA, 463

- Image, 46, 81, 82
 applications, 437
 bounding box, 417
 grey value, 417
 interpretation, 425
 knowledge containers, 431
 levels, 414
 mapping, 426
 processing, 418
 semantic interpretation, 427
 simplification, 417
 standards, 436
 texture, 417
 understanding process, 425
- Image computation
 challenges, 415
- Image processing, 424
- Implementation, 257
- Influence diagram, 101
- Information entities, 322, 397
- Information extraction, 396
- Information retrieval, 42, 376, 391, 525
- Information theory, 516
- Iriondo I, 463
- J**
- Jain R, 147
- Jantke K, 245
- Japkowicz N, 273
- Jaulent MC, 442
- Jensen–Shannon, 360
- Jiang–Conrath measure, 400
- K**
- Kamp G, 84
- kd-trees, 178–183
- Keane MT, 219, 298
- Khator S, 338
- Kibler D, 245
- Kim E, 485
- Kim JG, 245
- Kim JW, 246
- King JA, 187
- KM, 13, 75–77, 487, 523
 cycle, 492, 493
 discipline, 490
 goals, 491
 problems, 489
 repository-based, 499
- Knowledge, 488
 unit, 497
- Knowledge acquisition, 531
- Knowledge-based systems, 34, 524, 530
- Knowledge bases, 530
- Knowledge containers, 31, 34–37, 39, 42, 142, 252, 280, 284, 287, 294, 467
 evaluation, 251
 filling, 255
 maintenance, 263
 relations, 263, 277
 shifting, 279
- Knowledge distribution, 498
- Knowledge infrastructure and organisation, 493
- Knowledge-intensive, 281
- Knowledge management, *see* KM
- Knowledge model, *see* Knowledge containers
- Knowledge organisation and retrieval, 494, 495
- Knowledge representation, 530
- Knowledge retrieval and reuse, 495
- Knowledge sharing, 497
- Kockskämper S, 84
- Kolodner JL, 16, 40, 219, 538
- Kontkanen P, 245
- Koychev I, 409
- Kullback–Leibler, 359, 400
- L**
- Lamontagne L, 408
- Lange S, 245
- Langseth H, 245, 372
- Lapalme G, 408
- Latent semantic indexing, 388–390
- Law, *see* Legal
- Le Bozec C, 442
- Leacock–Chodorow measure, 399
- Leake DB, 219, 273, 298, 318
- Learning, 226–228
 eager vs. lazy, 280
- Learning of similarity relations, 235
- Learning with feedback, 234
- Learning without feedback, 234
- Legal, 78–80
- Lenz M, 52, 338, 408
- Leonardi G, 463
- Lessons learned, 497
- Lester JC, 407, 485
- Levenshtein measure, 133, 151
- Li H, 219
- Li J, 84
- Li X, 219
- Lieber J, 219, 245
- Lin measure, 399
- Linear predictive coefficient, 448
- Linguistics, 381
- Links, 108
- Local-global principle, 9, 10, 62, 96, 100, 109, 110, 114, 133, 143, 145, 157, 158, 169,

- 179, 253, 302, 305, 307, 310, 311, 360, 378, 426, 433, 462, 510
- Logic
 - reasoning in, 514
- Logic vs. logic formalism, 510
- Logical inference, *see* Logical reasoning
- Logical reasoning and similarity, 315
- Lopez B, 52
- López de Mántaras R, 245
- Lothian R, 338, 409
- M**
- MAC/FAC, 172, 173, 205
- MacDonald C, 298
- Machine learning, 56, 104, 143, 238, 251, 305, 523, 531, 532
- Macura K, 442
- Macura R, 442
- Maier V, 463
- Maintenance
 - actions, 261
 - indicators, 262
 - operations, 263
 - participants, 268
 - periphery, 270
 - reactions, 261
 - team, 268
- Mamdani, 345
- Manago M, 186
- Maney T, 484
- Many-valued attributes, 158
- Marling C, 15, 84
- Martins A, 408, 442
- Massie S, 409
- Matwin S, 273
- McCarthy K, 485
- McClave P, 52, 338
- McGinty L, 298, 485
- McKenna E, 220
- McSherry D, 245, 338, 485
- Measurement data
 - processing steps, 445
- Mel-frequency Cepstral coefficient, 455
- Messmer BT, 164
- MFCC features, 455
- Mille A, 219
- Minkowski distance, 130
- Minor M, 164
- Missing values, *see* Unknown values
- Model, 512
- Modus ponens, 515
- Moen H, 408
- Monotonicity, 307
- Montaner M, 52
- Montani S, 463
- Monzo C, 463
- Moore RK, 463
- MOPs, 5, 533
- Mott B, 485
- Mougouie B, 319
- Muñoz-Avila H, 164, 219, 485, 504
- Myllymaeki P, 245
- N**
- n-grams, 386
- Napoli A, 219
- Nearest neighbour, 29, 117, 118, 120, 139, 169, 171, 173, 175, 181, 202, 231, 232, 236, 299, 314, 334, 335, 341, 434, 450, 456, 529
- Necessity, 354
- Negation, 379
- Neural networks, 239
- Noise, 228, 232, 445, 446
- O**
- O'Sullivan D, 52
- Object-oriented representation, 96–102
- Object-oriented similarities, 156–158
- Okamoto S, 246
- Ontologies, 285–288, 392, 480
- Opening operator, 428
- Opposites, 379
- Optimization, 520
- Organisation-oriented applications, 70
- Oroumchian F, 246
- Overfitting, 228
- Öztürk P, 408
- P**
- Pacheco RC, 442
- Paikari E, 246
- Park SC, 246
- Path-oriented measures, 400
- Pattern recognition, 527–529
- Periphery, 256
- Perner P, 441, 442
- Planning, 20, 66–68
- Plans, 105, 106
- Plaza E, 15, 40, 164, 220, 298, 503
- Polygon approximations, 420
- Polymorphism, 383
- Polysemy, 383
- Porenta G, 441
- Porter B, 338
- Portinale L, 463
- Possibility theory, 352
- Prasad MV, 298
- Prasath R, 408

- Predicate logic, 511
 - Prediction, 20, 55, 62–64
 - Predictive power, 144
 - Probability
 - dynamics, 366
 - influences, 365, 366
 - Problem solving
 - model, 488
 - Process, 107
 - stochastic, 368, 449
 - Process model, *see* CBR cycle
 - Product base, 42, 48, 168, 198, 199, 203, 204, 360
 - Propositional logic, 510
 - PROTOS, 330
 - Prototype, 251
 - Provenance, 6, 11, 67, 225, 265, 290, 294, 380, 404
- Q**
- Question-answer, 494
 - Question selection, 475
- R**
- Racine K, 485
 - Radeva P, 246
 - Rahgozar M, 246
 - Random indexing, 390
 - Reachable, 206
 - Recio-García JA, 40, 273, 298, 338, 408
 - Recommender systems, 6, 42, 46
 - Reduced representations, 387
 - Redundancy, 313
 - Regression learning, 239
 - Reilly J, 485
 - Reinforcement learning, 241
 - Relevance-oriented measures, 397
 - Reliability, 380, 536
 - Repair, *see* Revision
 - Replay, 163, 214, 215, 218
 - Representation, *see* Case representation
 - Representation layers, 87
 - Resnik measure, 399
 - Retrieval, 26–30, 167–169, 451
 - approximation, 176
 - errors, 169
 - filtering, 170
 - fuzzy, 331
 - index-based, 177
 - methods, 170
 - sequential, 171
 - two-level, 172
 - Reuse, 30, 31
 - Revise, *see* Revision
 - Revision, 221–226, 260
 - Richter MM, 16, 40, 52, 84, 147, 246, 298, 441
 - Riesbeck CK, 16
 - Risk analysis, 360
 - Roth-Berghofer TR, 273, 319
 - Rough sets, 231, 340–342
 - Ruhe G, 84, 147, 246
 - Rules, 191, 514
 - actions, 192
 - adaptation, 196
 - completion, 193
 - of inference, 514
 - preconditions, 192
- S**
- Sakakibara Y, 245
 - Salamó M, 246
 - Santini S, 147
 - Satoh K, 246
 - Schaaf J, 338
 - Schank RC, 16, 538
 - Schmitt S, 52, 485
 - Schulz S, 164
 - Schumacher J, 40, 187
 - Search space, 332
 - Segmentation block, 454
 - Semantic ambiguity, 383
 - Sensor data
 - applications, 459
 - feature extraction, 444
 - levels, 445
 - Sequence, 105, 106
 - Sets, 107
 - Shewchuk JR, 84
 - Shimazu H, 485
 - Shin K, 246
 - Shiu SCK, 355
 - Silander T, 245
 - Similar, *see* Similarity
 - Similarity, 21, 26–30
 - axioms, 306
 - between attributes, 27
 - computation complexity, 310
 - distances and, 122–124
 - first- and second-order, 308
 - formal aspects, 300
 - foundations, 299
 - function, 120–122
 - fuzzy equality, 115
 - global, 302
 - goal, 27
 - group, 152
 - jump, 312
 - local, 302

- Similarity (*cont.*)
 - meaning, 115
 - metrics, 115
 - noise, 312
 - path, 153
 - processes, 159
 - relation, 116
 - time series, 161
 - workflows, 159
- Similarity and explanations, 314
- Similarity container, 35, 42, 234, 253, 277, 278, 315, 397, 431, 450, 466, 469
 - maintenance, 264
- Similarity measure, 49, 61
 - local, 135–138
 - path, 400
 - probabilities, 358
 - query to text, 402
 - segments, 400
 - statistics, 358
 - structure-oriented, 130–133
 - text-to-text, 401
 - transformational, 133
 - types, 125
 - weighted, 134
- Skeletons, 423
- Smyth B, 15, 52, 219, 220, 273, 298, 338, 485
- Socoró JC, 463
- Sooriamurthi R, 298
- Spam filtering, 80, 81
- Spatial position description, 421
- Speech
 - applications, 459
 - cycle, 455
 - features, 444
 - generation, 452
 - process, 452
 - recognition system, 454
 - semantics, 457
 - speech recognition, 451
 - understanding, 456
- Stahl A, 164, 246, 319
- Standard CBR view, 41
- Stop words, 384
- Stottler RH, 187
- Subjectivity, 304
- Syntactic parsing, 384
- Synthetic tasks, 53
- System development
 - steps, 249
- T**
- Taha D, 84
- Tartakovski A, 164
- Tautology, 513
- Tautz C, 273, 504
- Taxonomic similarities, 153
- Taxonomy, 100
- Testing, 269
- Text, 80, 81, 375
 - levels, 378
 - preprocessing, 384
 - problems understanding, 382, 383
 - properties, 380
 - semistructured, 376
 - similarity, 376
 - unstructured, 376
- Text mining, 376
- Textual CBR, 375
- Thesauri, 378, 392, 479
- Threshold operator, 428
- Tirri H, 245
- Tokens, 386
- TOPs, 534
- Traphöner R, 441
- Tree structures, 286
- Trees, 104
- Triangle inequality, 327
- Tversky measure, 128, 306
- U**
- Uncertainty, 339, 535
 - reasons, 339
- Unknown values, 313
- User-independent conditions, 293
- User interfaces, 256
- User model, 473
- Utility, 300, 518
 - expected, 518
 - theory, 518
- V**
- Value difference metric, 358
- Vazifedoost AR, 246
- Vector representations, 387
- Veloso MM, 84, 220
- Virtual attribute, 141
- Vocabulary container, 35, 76, 229, 252, 277, 316, 384, 431, 456, 460, 468
 - maintenance, 263
- von Neumann-Morgenstern
 - axioms, 518
 - utility representation theorem, 519
- Voronoi diagrams, 173
- W**
- Wang XZ, 338
- Watson ID, 298, 504

- Weber RO, 84, 297, 298, 338, 355, 408, 409, 442, 504
- Weights, 28, 138–140
diversity, 309
learning, 236
- Weir CC, 338
- Wess S, 52, 111, 186, 187, 245
- Whitehead M, 273, 298
- Whitehouse P, 84
- Wilke W, 52, 111, 272, 504
- Wilkinson L, 338
- Wilson DC, 52, 219, 273, 409, 441
- Wiratunga N, 298, 338, 408, 409
- Wombacher A, 165
- Word sense disambiguation, 394
- WordNet, 393
- Workflows, 107
- Wu–Palmer measure, 153
- Y**
- Yang Q, 273, 485
- Yeung DS, 338
- Z**
- Zaluski M, 273
- Zapletal E, 442
- Zimring C, 441