知識グラフに基づく質問応答における論理形式パターンを用いた論理形式検索の高速化

Effective Search of Logical Forms in Question Answering based on the Knowledge Graph using Logical Form Patterns

吉兼　拓生 [1]　谷津　元樹 [2]　森田　武史 [2]

Takumi Yoshikane [1], Motoki Yatsu [2], and Takeshi Morita [2]

[1] 青山学院大学大学院 理工学研究科

[1] Graduate School of Science and Engineering, Aoyama Gakuin University

[2] 青山学院大学 理工学部

[2] College of Science and Engineering, Aoyama Gakuin University

Abstract: CSQA is a question answering benchmark using the knowledge graph, that created using the large-scale knowledge graph database Wikidata. CSQA targets interactive question answering, and there are 10 question types, some of which require inference such as comparison and set operation. In the previous study, multi-task semantic parsing model that converts utterances to logical forms that expresses operations on a database can answer high accuracy. The conversion to a logical form is performed by defining the grammar for deriving the logical form and predicting the order in which the grammar is applied from the utterance. In order to learn a model for the prediction, it is necessary to search for a logical form that can answer the question from logical forms that can be generated by applying the grammar. However, since the search space is enormous, depending on the search method, problems such as a decrease in the success rate of the search and incorrect logical forms occur, which may adversely affect the learning of the model. In this research, we propose a method for searching logical forms with a high success rate of the search in a short time by using patterns of logical forms.

1. はじめに

知識グラフに基づく質問応答の研究では，モデル学習の前処理として，質問文の意味を表す論理形式を導出するための文法を定義し，文法に基づいて論理形式を生成し，その中から質問に回答可能な論理形式を検索している．推論を必要とする複雑な質問に対応する論理形式を生成可能な文法は，巨大な検索空間を生み出し，検索時間やメモリ使用量のコストが大きくなるという問題がある．しかしその問題への対処の仕方によっては検索成功率が低い，意味的に誤った論理形式が検索されるなどの問題が発生する．

大規模な知識グラフ Wikidata を用いた対話型質問応答データセットである CSQA[1]に探索幅を制限した幅優先探索を適用して探索空間を小さくして論理形式の検索を行った研究[2, 3]では，比較に関する質問と量に関する質問に対する論理形式の検索成功率がそれぞれ 25%と 43%にとどまるほか，誤った論理形式が 54.5%検索されることが報告されている[4].

質問ごとに使用すべき文法を予測することで検索空間を小さくする研究[4]では，[2, 3]に適用した結果検索成功率が上昇し，誤った論理形式の割合が減少し，論理形式が検索できた質問 1 問辺りにかかる時間が減少し，質問応答自体の精度も上昇したことが報告されている．

しかし，[4]においても検索成功率が 40%に満たない質問タイプが存在する，意味的に誤った論理形式が 26.7%存在するという問題がある．

他の論理形式検索に関する研究として, 表に対する質問応答データセットで論理形式検索の高速化を行った研究[7]では，論理形式のパターンなどを利用することで論理形式検索の高速化を図っている．

また, CSQA に対して論理形式を生成して回答を行うフレームワーク[2, 3, 5, 6]では，一旦論理形式のパターンを生成した後にその詳細を決定して論理形式を生成している．このような形でうまく学習が行えているのは，複数の質問の間で共通する論理形式の

パターンがあり，それをうまくとらえることができているためではないかと推測できる．

本研究では，知識グラフに基づく質問応答システムにおけるモデル構築の前処理に必要な論理形式検索を，論理形式のパターンを用いて高速化するための手法を提案する．

2. 関連研究

[2]は論理形式の生成を連続するアクション（適用する文法）の予測と対応付け，Encoder-Decoder モデルで質問文から文法の適用順序を予測することで論理形式を求めるフレームワーク Dialog-to-Action (D2A)を提案している．D2A は CSQA データセットにより評価が行われている．

D2A は，GRU[8]を用いた双方向 RNN を Encoder として用い，アテンション機構を備えた GRU を Decoder として用いる．Decoder の出力時点では知識グラフ中のエンティティなどの要素の決定は行わず，後述するダイアログメモリにより決定する．現在の質問に含まれるエンティティなどの他，以前の質問や回答に出現したエンティティや，(木構造として表される）論理形式の部分木を管理するダイアログメモリを導入し，対話型質問応答で発生する代名詞による参照や語の省略に対処する．論理形式の導出のための文法は独自に定義している．学習用の論理形式の検索には探索幅を制限した幅優先探索を使用している．

[3]は D2A のようなエンティティ検出などの上流タスクや質問文の論理形式へのマッピングなどの下流タスクを別々に学習する段階的なアプローチでは，上流タスクのエラーが下流タスクに伝播して蓄積したり，各タスク用のモデル間で教師信号を有効に活用できなかったりするという点を問題として指摘している．その問題を解決するために，上流タスクと下流タスクのマルチタスク学習を行えるようにしたフレームワーク **M**ulti-t**a**sk **S**emantic **P**arsing (MaSP)を提案している．Transformer[9]を使用した Encoder と Decoder を採用しており，Decoder からの出力に対するエンティティの決定には pointer network[10]を使用している．論理形式の導出用の文法は D2A と同様のものを使用しており，学習用の論理形式の検索も D2A 同様に探索幅を制限した幅優先探索を使用している．

[4]は，CSQA において文法を用いた論理形式の際に D2A や MaSP で使用された探索幅を制限した幅優先探索では比較に関する質問と量に関する質問に対する論理形式の検索成功率がそれぞれ 25%と 43%にとどまるなど，検索成功率が低い場合があることを示している．また，仮に論理形式が見つかったとしても偶然回答にたどり着いただけで意味的には誤った論理形式である場合があり，単純に探索幅を制限した幅優先探索を行った場合には人手での評価で 54.5%の論理形式が誤った論理形式であると示している．[4]では，一部の質問に対する検索結果から質問に対して使用すべき文法を予測する予測器を学習し，各質問に対する探索空間を小さくすることにより高い検索成功率で検索を行うほか，誤った論理形式を 26.7%まで減少させた．また，D2A と MaSP に適用した結果，質問応答の精度も改善することを示している．

以下では[4]を論文の題目から”Effective Search”と呼ぶ．

[5]は，MaSP ではエンティティ検出のモデルが Encoder の信号のみを使用し，Decoder などの信号を利用できていないと指摘している．MaSP ではエンティティの決定にのみ Pointer Network を用いていたが，[5]では関係とタイプの選択にも用いるために知識グラフの埋め込みを用いる 3 つの Pointer Network を使用するフレームワーク **C**ontext tr**A**nsforme**R** s**T**acked p**O**inter **N**etworks (CARTON)を提案している．Encoder と Decoder は MaSP と同様に Transformer を使用している．論理形式の導出用の文法は MaSP の文法に新しい文法を追加して拡張したようなものを使用する．論理形式の検索は質問タイプなどにより場合分けを行い，その場合ごとにあらかじめ人手で決めた論理形式のパターンにエンティティなどを当てはめることで行う．

[6]は，MaSP では関係とタイプの決定を行うモデルが互いに信号をやり取りしておらず，知識グラフ上に存在しない組み合わせの関係とタイプを出力する可能性が存在すると指摘している．[6]では，graph attention networks(GATs)[11]により知識グラフ中で関連している関係とタイプを選択できるようにするフレームワーク mu**L**ti-task sem**A**ntic par**S**ing with tr**A**nsformer and **G**raph atte**N**tion n**E**tworks (LASAGNE)を提案している．論理形式の導出用の文法は MaSP の文法に CARTON とは別の新しい文法を追加して拡張したようなものを使用する．Encoder と Decoder は MaSP と同様に Transformer を使用している．論理形式の検索は CARTON と同様に質問タイプなどにより場合分けを行い，その場合ごとにあらかじめ人手で決めた論理形式のパターンにエンティティなどを当てはめることで行う．

[7]は，それまでの検索で見つかった論理形式とその部分木に対応するパターンを文法に追加することで論理形式の高速化を行う手法である．単純に文法を追加していくと文法が増えて検索空間が増大し

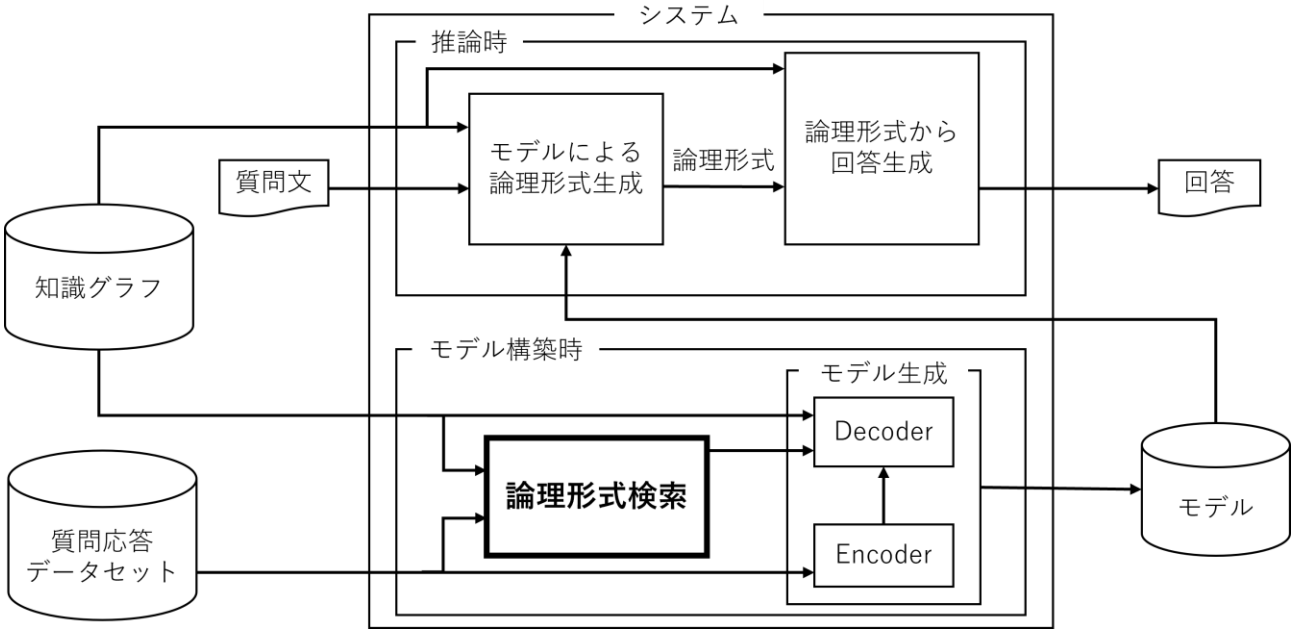

図 1 知識グラフに基づく質問応答システムの概念図

てしまうため，現在検索を行っている質問に類似している質問をレーベンシュタイン距離により求め，その質問から生成された文法のみを追加することで検索空間が大きくなりすぎないようにしている．

本研究では，D2A や MaSP で行われている学習用の論理形式検索の部分において，一部の質問に対する検索結果から質問に回答可能な論理形式のパターンを求め，そのパターンをもとに論理形式検索を行うことで探索空間を削減することで高速化を試みている．

3. 知識グラフに基づく質問応答データセット

CSQA は知識グラフに基づく対話型質問応答データセットである．約 20 万の対話形式の質問応答からなる大規模なデータセットであり，合計で約 160 万の質問応答ペアが含まれる．CSQA では知識グラフのトリプルを取得するだけで回答可能な単純な質問が 3 種類（通常，代名詞での参照あり，語の省略あり），Yes/No で回答が可能な質問，集合演算を必要とする質問，最大/最小などの量に関する質問が 2 種類（エンティティで回答，数値で回答），比較が必要な質問が 2 種類（エンティティで回答，数値で回答），聞き返しが必要な曖昧な質問の 10 の質問タイプが定義されている．

単純な質問は，以下の手順で作成する．まず初めにクラウドソーシングにより，知識グラフ中のトリプルの主語か目的語を尋ねる質問を作成する．次にその質問の言い換えを別のクラウドワーカーに依頼して，作成された質問をもとに質問のテンプレートを作成する．テンプレートから半自動的に質問を生成する．

比較や集合演算などが必要な複雑な質問は，組織内のアノテーターが単純な質問から作成するためのテンプレートやルールを作成し，それを利用することで作成する．

4. 知識グラフに基づく質問応答システム

4.1. システム構成

図 1 に知識グラフに基づく質問応答システムの概念図を示す．図の上側は実際に推論を行う際の流れを表している．質問を入力するとモデルが論理形式を生成して，その論理形式から回答を生成する．図の下側は推論時に論理形式の生成を行うモデルの構築の際の流れを表している．質問中に含まれるエンティティなどを用いて 4.1 節で示した例のような論理形式の導出を行い，実行した結果が質問に対する回答と一致する論理形式を検索する．一致するものが見つかれば，それをモデルの学習に利用する．
関連研究に挙げたフレームワークは，いずれもエンコーダーデコーダー方式のモデルを採用している．エンコーダーには直前の質問と直前の回答，現在の

表 1 D2A で用いている文法

	文法				文法		
1	start	→	set	12	set	→	equal(set, r, num)
2	start	→	num	13	set	→	argmax(set, r)
3	start	→	bool	14	set	→	argmin(set, r)
4	set	→	find(set, r)	15	set	→	e
5	num	→	count(set)	16	e	→	constant
6	bool	→	in(e, set)	17	r	→	constant
7	set	→	union(set_1, set_2)	18	num	→	constant
8	set	→	inter(set_1, set_2)	19	set	→	$action_{i-1}$
9	set	→	diff(set_1, set_2)	20	num	→	$action_{i-1}$
10	set	→	larger(set, r, num)	21	bool	→	$action_{i-1}$
11	set	→	less(set, r, num)				

質問の 3 つを連結した文を入力する．直前の質問と直前の回答も入力するのは，対話型の質問では以前出た語の省略や代名詞による参照などが発生することがあり，元の語の情報も必要となるためである．デコーダーにはエンコーダーの隠れ状態を入力し，文法の適用順を出力する．ここではエンティティなどの具体的な値は決定せず，その先に接続している別のモデル（例：pointer network）で決定を行う方式が関連研究では取られている．最終的に生成された論理形式を実行した結果が質問の回答と一致するようにモデルの学習が行われる．

4.2. 論理形式検索

データセットにシステムで使用したい論理形式が存在しない場合には，システム側で論理形式を見つける必要がある．その際に，形式文法を用意して，開始記号から文法を適用していくことで論理形式を作成することがある．その際に，最も左の非終端記号を置き換えていくなどのルールを定めておくと，文法の適用順序と論理形式が 1 対 1 に対応するため，モデルにこの文法の適用順序を出力するように学習させる場合がある．関連研究では文法の適用順序を出力するモデルが使用されている．

表 1 に D2A で用いている文法を示す[1]. e はエンティティ，r は関係，num は数値を表す．constant は質問中に含まれる実際のエンティティなどの値であり，set はエンティティの集合を表す．$action_{i-1}$ はダイアログメモリが管理している論理形式またはその一部を複製することを表す．この文法を用いて論理形式を導出する例は以下のようになる．（“So Long Mr. Chumps”というエンティティと”director”という関係があると仮定する）

① start （開始記号）
② set （1 番を適用）
③ find(set, r) （4 番を適用）
④ find(e, r) （17 番を適用）
⑤ find(“So Long Mr. Chumps”, r)（18 番を適用）
⑥ find(“So Long Mr. Chumps”, “director”) （19 番を適用）

この例のように文法の左側の set などの非終端記号がなくなったものが論理形式となる．この論理形式は，“So Long Mr. Chumps”というエンティティと“director”という関係で繋がっているエンティティを取得するという操作を表す．

5. 既存研究における論理形式検索

5.1. D2A と MaSP における論理形式検索

D2A と MaSP では，探索幅を制限した幅優先探索を用いて論理形式の検索を行う．開始記号から文法を適用して候補を生成していき，探索幅が指定の値を超えた際にランダムで指定の幅まで枝を減らす．文法の連続などの制限による枝刈りも行い，探索空間を小さく保つ．

5.2. Effective Search における論理形式検索

Effective Search では，最初に一部の質問に対して D2A や MaSP と同様に論理形式の探索を行う．その後質問タイプごとに，その文法を使用しなくても検

[1] 公開されているコード（https://github.com/guoday/Dialog-to-Action）では変更が加えられ、MaSP と同じ文法を使用している

コード 1 論理形式パターンの検索

```
INPUT: questionDataList, timeOut
  OUTPUT: patternList

  patternList ← []
  FOR each questionData in questionDataList
    logicalForm ← BreadthFirstSearch(questionData, timeOut)
      IF logicalForm is found
        logicalFormPattern ← MakePattern(logicalForm)
        append logicalFormPattern to patternList
        FOR each questionData in questionDataList that is not searched
          IF logicalFormPattern can answer questionData
            remove questionData from questionDataList
          END IF
        END FOR
      END IF
    END FOR

RETURN patternList
```

索成功率の下がり方が閾値より低い，すなわちその文法があっても検索成功率があまり上がらない文法を求め，元の文法からその文法を除いたものをその質問タイプに対して用いる文法とする．質問タイプが未知の場合には，各質問タイプに使用される文法をもとにクラスタリングを行い，そのクラスターを質問タイプとする．ここまでで各質問に対して使用すべき文法が分かっているため，その組を用いて質問から使用すべき文法を予測する予測器を学習させる．予測器が学習できたらすべての質問に対して使用する文法を予測する．最後に各質問に対して予測された文法を用いて，すべての質問に対して論理形式を検索する．

5.3. CARTON と LASAGNE における論理形式検索

CARTON と LASAGNE では，質問タイプとサブタイプごとに場合分けを行い，それぞれの場合に応じてあらかじめ人手で論理形式のパターンを作成しておく．その後各質問の質問タイプとサブタイプに応じて，その場合の論理形式のパターンにエンティティなどを当てはめることで論理形式の検索を行う．

5.4. 既存研究における問題点

D2A, MaSP のアプローチでは，途中でランダムに探索範囲を削減してしまうため，正しい論理形式にたどり着くことができずに検索に失敗する可能性が特に複雑な質問で大きくなる．また，Effective Search によれば誤った論理形式が検索されてしまう割合も大きい可能性が示されている．

CARTON, LASAGNE におけるアプローチでは，あらかじめ人手で回答が可能な論理形式のパターンを作成する部分にコストがかかるほか，パターンとして作成したもののみに探索を行うため，それ以外のパターンで回答できるものがあったとしても検索されないという問題がある．

6. 提案手法

提案手法では，論理形式のパターンを発見して論理形式の検索を行う．以下の 3 ステップで検索を行う．

- ステップ① 一部の質問に対して論理形式を検索し，見つかった論理形式からパターンを求める
- ステップ② ステップ①で見つかったパターンを用いてすべての質問の論理形式を検索する
- ステップ③ 1 つの質問に対し複数の論理形式が検索された場合にどれを残すか選択する

以下ではまず論理形式のパターンについて説明した後，それぞれのステップについて詳しく見ていく．

6.1. 論理形式パターン

本研究における論理形式のパターンとは，論理形式中のエンティティや関係などを取り除き抽象化したものである．4.2 節で出てきた論理形式の例

コード 2 論理形式パターンを用いた論理形式の検索

```
INPUT: questionDataList, patternList
  OUTPUT: questionDataList

  FOR each questionData in questionDataList
    canAnswerLogicalForms ←[]
    FOR each pattern in patternList
      logicalForm ← GetCanAnswerLogicalForm(questionData, pattern)
      IF logicalForm is found
        append logicalForm to ccanAnswerLogicalForms
      END IF
    END FOR
    append canAnswerLogicalForms to questionData
  END FOR

RETURN questionDataList
```

コード 3 使用する論理形式の決定

```
INPUT: patternList, questionDataList
  OUTPUT: questionDataList

  FOR each pattern in patternList
    count number of questions that have logical form made from pattern
  sort patternList by the number counted above

  FOR each pattern in patternList
    FOR each questionData in questionDataList
      IF not decide which logical form use for questionData
        IF questionData has logical form that made from pattern
          decide to use that logical form for questionData
        END IF
      END IF
    END FOR
  END FOR

RETURN questionDataList
```

find(“So Long Mr. Chumps”, “director”)のパターンは find(e,r)となる. パターンの抽象化した部分に別の質問のエンティティなどを入れることで，同じ構造の別の論理形式を作成することができる.

6.2. 論理形式パターンの検索

ステップ①では，一部の質問から質問に回答することが可能な論理形式のパターンを求める．コード 1 にステップ①の擬似コードを示す.

一部の質問を入力とし，各質問に対して幅優先探索で論理形式の検索を行う．回答可能な論理形式が見つかるか，指定の時間が過ぎてタイムアウトするまで検索を続ける．回答が可能な論理形式が見つかった場合はその論理形式のパターンを求める．パターンが見つかるごとに，検索をまだ行っていない質問に対してそのパターンで回答ができるかどうかを確かめ，回答できる場合はその質問を検索候補から外すことで既知のパターンで回答できない質問に時間をかけることができるようにする.

6.3. 論理形式パターンを用いた論理形式の検索

ステップ②では，ステップ①で求めた論理形式パターンを使用してすべての質問に対する論理形式の検索を行う．コード 2 にステップ②の擬似コードを

表 2 各質問タイプ 5000 問ずつで計測した際の検索成功率の比較

質問タイプ	D2A, MaSP	提案手法	Effective Search
Simple Question (Direct)	96.42%	97.58%	97%
Simple Question (Coreferenced)	91.54%	92.74%	
Simple Question (Ellipsis)	94.44%	97.54%	
Logical Reasoning	49.20%	**99.62%**	89%
Verification	79.22%	**83.96%**	81%
Quantitative Reasoning	42.80%	**58.74%**	57%
Quantitative Reasoning (Count)	69.58%	**80.64%**	
Comparative Reasoning	26.14%	**46.76%**	39%
Comparative Reasoning (Count)	34.48%	**53.94%**	
Clarification	**1.18%**	**1.18%**	-

表 3 各質問タイプ 5000 問ずつで計測した検索時間の比較

質問タイプ	D2A, MaSP	提案手法
Simple Question (Direct)	1 時間 14 分	**3 分 21 秒**
Simple Question (Coreferenced)	1 時間 22 分	**8 分 20 秒**
Simple Question (Ellipsis)	3 時間 15 分	**3 分 11 秒**
Logical Reasoning	7 時間 57 分	**1 時間 39 分**
Verification	5 時間 53 分	**28 分 19 秒**
Quantitative Reasoning	**38 分 31 秒**	1 時間 16 分
Quantitative Reasoning (Count)	2 時間 20 分	**39 分 59 秒**
Comparative Reasoning	**1 時間 41 分**	1 時間 51 分
Comparative Reasoning (Count)	1 時間 53 分	**1 時間 38 分**
Clarification	**2 時間 23 分**	2 時間 49 分

示す.

ここでは探索範囲をステップ①で求めたパターンに各質問のエンティティなどを当てはめて作成できる論理形式のみに制限して検索を行う．ステップ②では論理形式が見つかったとしても，その質問とそのパターンの組み合わせでの検索を終了するだけで，すべての質問とパターンの組み合わせでの検索を行う．

6.4. 使用する論理形式の決定

ステップ③では，各質問で検索された論理形式のうちどれを使用するかを決定する．コード 3 にステップ③の擬似コードを示す.

ステップ②で使用したすべてのパターンについて，何問の質問に対して回答可能な論理形式を検索することができたかを数える．その回答可能な論理形式の数が多いパターンから作成された論理形式を優先して，各質問の論理形式を決定する.

7. 評価

7.1. 検索成功率

表 1 は質問タイプごとに 5000 問ずつで検索を行った際の検索成功率の比較である．提案手法では最初に 500 問で論理形式のパターンを検索した後に質問全体に対して論理形式の検索を行っている．この論理形式の検索時には，D2A と MaSP と同様の枝刈りとタイムアウト時間（20 秒）を使用した．すべての質問タイプで D2A と MaSP の方式を提案手法での検索成功率が上回っている．Effective Search の数値は手元で実行した値ではなく論文中に記載の値であり，同系統の質問はまとめられてしまっているが，Logical Reasoning，Verification で提案手法の検索成功率が上回り，Quantitative Reasoning 系統，Comparative Reasoning 系統もまとめた場合での検索成功率は提案手法が上回っている.

7.2. 検索時間

表 2 は質問タイプごとに 5000 問ずつで検索を行った際の検索にかかる時間の比較である．提案手法は 500 問での論理形式パターンの検索を行っている．Simple Question，Logical Reasoning，Verification，Quantitative Reasoning (Count)，Comparative Reasoning (Count)で D2A, MaSP よりも検索時間が減少した．

7.3. 考察

提案手法で検索成功率が既存手法よりも上回ったのは，発見されたパターンをすべての質問に適用したことで，既存手法ではそのパターンで回答できるにもかかわらずランダムな枝の削減により検索が行われなかった質問に対しても検索を行うことができたためと考えられる．

提案手法で既存手法よりも検索時間が短くなったのは，既存手法と同じかそれ以上の検索空間で検索を行う質問は一部にとどまり，見つけたパターンでのみ検索を行うことで検索空間が小さくなった質問が多かったためと考えられる．

上記 2 つの結果が得られたのは，検索空間に対してごく一部の論理形式パターンで多くの質問に回答が可能であるためと考えられる．

8. おわりに

本研究では，知識グラフに基づく質問応答システムにおけるモデル構築の前処理に必要な論理形式検索を，論理形式のパターンを用いて高速化する手法を提案した．一部の質問タイプで既存手法よりも検索成功率を増加させ，検索にかかる時間を減少させることができた．今後は誤った論理形式の割合や，実際の質問応答精度に与える影響及び，CARTON などの人手でパターンを求める場合との比較などの調査を行う予定である．

参考文献

[1] Saha, A., Pahuja, V., Khapra, M. M., Sankaranarayanan, K., & Chandar, S.: Complex Sequential Question Answering: Towards Learning to Converse Over Linked Question Answer Pairs with a Knowledge Graph. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pp. 705–713. (2018)

[2] Guo, D., Tang, D., Duan, N., Zhou, M., & Yin, J.: Dialog-to-Action: Conversational Question Answering Over a Large-Scale Knowledge Base, *NIPS'18: Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2018*, pp. 2946-2955. (2018)

[3] Shen, T., Geng, X., Qin, T., Guo, D., Tang, D., Duan, N., Long, G., & Jiang, D.: Multi-task learning for conversational question answering over a large-scale knowledge base. *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, pp. 2442–2451. (2020)

[4] Shen, T., Geng, X., Long, G., Jiang, J., Zhang, C., & Jiang, D.: Effective search of logical forms for weakly supervised knowledge-based question answering. *IJCAI International Joint Conference on Artificial Intelligence, 2021-Janua*, 2227–2233. (2020).

[5] Plepi, J., Kacupaj, E., Singh, K., Thakkar, H., & Lehmann, J.: Context Transformer with Stacked Pointer Networks for Conversational Question Answering over Knowledge Graphs. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *12731 LNCS*, 356–371. (2021).

[6] Kacupaj, E., Plepi, J., Singh, K., Thakkar, H., Lehmann, J., & Maleshkova, M.: Conversational Question Answering over Knowledge Graphs with Transformer and Graph Attention Networks. *EACL 2021 - 16th Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference*, 850–862. (2021).

[7] Zhang, Y., Pasupat, P., & Liang, P.: Macro Grammars and Holistic Triggering for Efficient Semantic Parsing. *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, 1214–1223. (2017).

[8] Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y.: Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 1724–1734. (2014).

[9] Jones, H., Moore, J., & Kenyon, G.: *Attention Is All You Need*. (2019).

[10] Vinyals, O., Fortunato, M., & Jaitly, N.: Pointer Networks. *Advances in Neural Information Processing Systems*, *2015-January*, 2692–2700. (2015).

[11] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y.: Graph Attention Networks. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*. (2017).