Accelerating the Annotation of Lexical Data for Less-Resourced Languages

Gerhard B van Huyssteen & Martin J Puttkammer

Centre for Text Technology (CTexT), North-West University, Potchefstroom, South Africa

{Gerhard.VanHuyssteen; Martin.Puttkammer}@nwu.ac.za

Abstract

The development of digital resources is an expensive and time-consuming endeavor, especially in the case of less-resourced languages. In this paper, we describe a freely available, open-source system, called *TurboAnnotate*, for bootstrapping linguistic data for machine-learning purposes, or for manually creating gold standards or other annotated lists. A detailed description of the design and functionalities of the tool is given, focusing on how the requirements of end-users are being addressed through it. It is indicated that *TurboAnnotate* does not only promise to help increase the accuracy of human annotators, but also to save enormously on human effort in terms of time.

Index Terms: hyphenation, bootstrapping, less-resourced languages, Afrikaans, Setswana

1. Introduction

Since the development of human language technologies (HLTs) depends in general on the availability of linguistic data, such as specialized lexicons, annotated and raw corpora, or formalized grammar rules, the development of such resources are imperative for the HLT enablement of any language. However, it is a well-known fact that the creation of such resources is an expensive and protractive endeavor, especially in the case of less-resourced languages.

In general, less-resourced (a.k.a. resource-scarce) languages are defined as "languages for which few digital resources exist; and thus, languages whose computerization poses unique challenges. [They] are languages with limited financial, political, and legal resources… " [1].

Implicit in this definition is the fact that low-resourced languages also often lack human resources, but this aspect seldom receives the necessary attention in research or discussions. Computational linguists working on these languages, or linguists and mother-tongue speakers with an inclination or the means to computerize these languages (i.e. to create digital resources), are often hard to find – even more so in developing countries where other pressing issues determine the priorities of skilled people. The question thus arises how one could effectively facilitate the development of linguistic data by enabling non-experts to collaborate in the computerization of less-resourced languages.

In this project, we investigate ways to create user-friendly environments (i.e. software with graphical user interfaces - GUIs) and/or methods (such as bootstrapping) for escalating the annotation of linguistic data by mother-tongue speakers with little or no experience in computational linguistics. Instead of using rule-based methods for the development of re-usable core technologies (such as hyphenators, lemmatizers, stemmers, etc.), we rather use machine learning techniques. The rationale behind this is to rather spend time and money on developing re-usable, annotated data for these languages, than to invest in methods that require deep linguistic and/or specialized knowledge (e.g. knowledge about regular expressions or finite-state automata). Such annotated data could, for instance, also be seen as part of an enriched lexicon that would be a "central aspect of any [natural language processing] project" [2].

For purposes of this article, we focus on our freely available, open-source system, called *TurboAnnotate* (version 1.0.0), for the development of gold standards and annotated data through bootstrapping. Our attention in this first version is specifically on the development of hyphenated data for two less-resourced South African languages (Afrikaans and Setswana). In Section 2 we discuss some general points of departure, based on our user-requirement analyses. Section 3 describes the system in detail, while Section 4 presents some results. This article concludes with a discussion of future possible extensions of the system.

2. End-user requirements

Our central point of departure in this project is that, for any given language, annotators (e.g. linguists, mother-tongue speakers with a linguistic sensitivity, or student assistants) are invaluable resources towards the creation of digital resources for that language. Based on our experiences in the past with less-resourced languages, it is often the case that such annotators have mostly word processing skills in a GUI-based environment, and not necessarily advanced skills in a computational or programming environment. In worst cases, annotators sometimes have difficulties with file management, unzipping, proper encoding of text files, etc. The aim of this project is thus to maximize their experience, by enabling them to focus on what they are good at: enriching data with their expert linguistic knowledge.

In order to determine how we could enable annotators (i.e. determine end-user requirements) we conducted unstructured interviews with four annotators who are currently working on some of our data annotation projects (usually working in simple text editors or spreadsheet programs). We asked two basic questions: (1) What do you find unpleasant about your work as an annotator?; and (2) What will make your life as an annotator easier?

The answers to question (1) are summarized as follows:

- All of them find the repetitiveness of annotation work rather tedious and boring. They suffer to concentrate for longer than thirty minutes at a time, and have to motivate themselves constantly to get back to the "laborious task of looking at one word after the other".
- In addition, they all experience physical strain on their eyes, necks, arms/elbows and backs. One respondent described annotation work as "backbreaking and physically demanding".
- Although they have a passion for linguistics and "enjoy working with language", they often feel "useless" because they don't "see the bigger picture". Also related to this, is the fact that they don't see results: "We send

our work off to the developers, only to hear weeks or months later what the results were."

With regard to question (2), the respondents described their ideal working environment as follows:

- They want to work in a "friendly" environment (i.e. GUI-based, and not lists of words).
- Since annotation is such a tedious task, they rather prefer to work in chunks, finishing off "bite-sizes of data" rather than "endless lists".
- They find it easier and quicker to correct data (i.e verification), than to annotate from scratch. All annotators agreed that it would be helpful if a program could already suggest a possible annotation, only for them to verify whether the annotation is correct or not.
- They just want to click or drag: "…no moving around with arrows, or inserting symbols with difficult hand-eye-coordination".
- Reference works (such as Google, online dictionaries, etc.) need to be readily available – preferably without starting up various other applications.
- A tool or mechanism that would automatically manage their work, such as version control, saving, etc.

The above dislikes and preferences were analyzed and rendered into end-user requirements, which steered the design and development of our annotating system, called *TurboAnnotate*.

3. Solution: *TurboAnnotate*

TurboAnnotate is a user-friendly annotating environment (i.e. tool) for bootstrapping linguistic data for machine-learning purposes, or for manually creating gold standards or other annotated lists. In as such, the design of *TurboAnnotate* was inspired by two other annotating tools, viz. *DictionaryMaker* [3] and *Alchemist* [4].

DictionaryMaker is a pronunciation dictionary bootstrapping system, allowing "a speaker fluent in the target language to develop a pronunciation dictionary without requiring expert linguistic knowledge or programming expertise" [3]. The backbone of the bootstrapping system is the *Default&Refine* algorithm, which is used for rule extraction [5]. Positive results are reported in [6], indicating "[f]or a 10,000 word dictionary, the bootstrapping approach requires 23% of the effort of the manual approach."

Alchemist is a GUI-based tool, designed to create morphological gold standards for text data [7]. While it was initially conceptualized as a tool to create data for machine learning purposes, its wider applicability for field linguists and in teaching was soon realized. In a computational linguistic environment, "*Alchemist* increases the efficiency of the human researcher and shortens the time of creating grammars, interlinear texts, and dictionaries" [7].

Drawing on the experiences of the aforementioned programs, as well as our end-user requirements, we developed a first version of *TurboAnnotate*, specifically with the task of hyphenation for South African languages in mind.

Essentially, *TurboAnnotate* implements three steps, which are next discussed in more detail (see Figure 1).

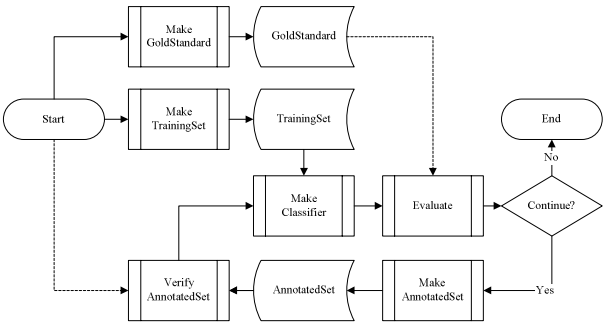

Figure 1: Simplified workflow of *TurboAnnotate*

3.1. Step 1: Create gold standard

After a new project has been created, the first step is to create a gold standard – i.e. an independent test set that will be used for evaluating the performance of classifiers. Note that an annotator only has to select one file (the data file/base list); the rest of file management is handled by the tool (see Figure 2). A random list of instances for the gold standard is automatically extracted from a base list (not indicated in Figure 1). The size of the gold standard is by default set on a 1,000 words, but could be changed under the "Options" tab. Although this might seem like a relatively small evaluation set, it was decided to keep it rather small in order to make it more manageable for annotators. Moreover, an exploratory experiment with a training set of 39,500 hyphenated Afrikaans words indicated an accuracy of 99.20% on an evaluation set of 2,200 words, and 99.60% accuracy on an evaluation set of 1,000 words. It thus led us to believe that, at least for the hyphenation task, a gold standard of a 1,000 words would suffice, without presenting a completely skewed view of the actual performance of the system.

TurboAnnotate could, of course, also be used to create gold standards only, without including steps 2 and 3.

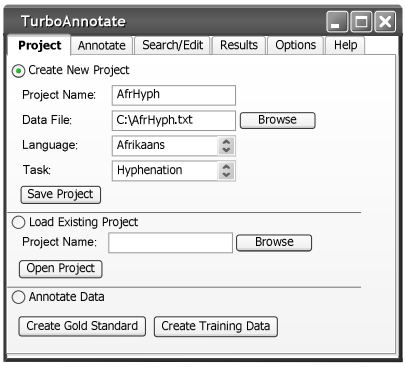

Figure 2: Screenshot: creating a project

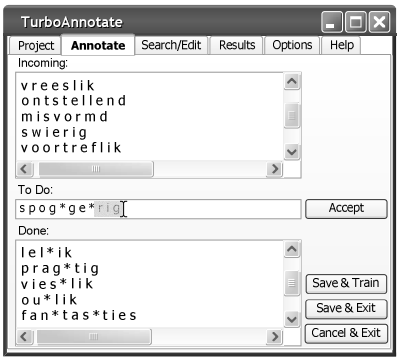

Figure 3: Screenshot: annotation environment

3.2. Step 2: Create training set

Before the bootstrapping process can commence (step 3), a training set to train a first classifier has to be created. Once again, based on the annotators request for "bite-sizes of data", we decided to experiment with very small sets of training data. The default size is therefore set on 200 words, which could also be changed under "Options". Data is being stored automatically both in annotated format (together with the original word-form, in two columns), and in a format suitable for the machine learning system.

The annotation GUI (see Figure 3) is to some extent inspired by *Alchemist*: the annotator simply drags the mouse over the part of the word to be annotated, and on release of the mouse button, the selection changes color, and an asterisk is inserted after the selection (in the case of data for hyphenation). Alternatively, there is also an option available to use the keyboard only for annotation. The annotator is also provided with the opportunity to immediately apply changes to words already being annotated (in the "Done" section), or to see words still to be annotated (in the "Incoming" section). In the "Search" tab (not shown here), h/she can also search for words in training sets already been annotated.

The machine learning system that we use in our system is the well-known Tilburg Memory-Based Learner (TiMBL; [8]). The choice for TiMBL is based on its wide success and applicability in the field of natural language processing, its availability for research purposes, as well as its relative ease to use. On the down-side, it is widely known that memory-based machine learning systems perform best with large quantities of data, which is an undesirable characteristic for purposes of developing data for less-resourced languages; however, it will be indicated that, at least for the task of hyphenation, TiMBL performs quite well, even with little data available.

The default parameter settings for TiMBL are automatically being used, while the selection of features is currently controlled by the system developers. Due to the relative complexity of these aspects of machine learning, it was decided to simplify matters for annotators, rather than to present them with a plethora of choices that will probably be more confusing than enabling.

After a classifier has been trained, it is evaluated against the complete gold standard to determine its accuracy. For hyphenation, accuracy is determined on word-level (i.e. all hyphens in a word), and not per correct instance; however, the user has the option to change this way of evaluation under the "Option" tab. Simplified results are displayed in the "Results" tab (not shown here), and the user has the choice to save data and exit, or to continue with the annotation process.

3.3. Step 3: Verify annotated set

If the annotation GUI was inspired by *Alchemist*, then this step was inspired by *DictionaryMaker*, since it represents the iterative bootstrapping phase in the design.

New data is sourced from the base list (not shown in Figure 1), in chunks determined under the "Options" tab (default is 200 words). This data is automatically annotated by the trained classifier (created in the previous step), and presented to the human annotator in the "Annotate" tab.

Next, the annotator has to verify whether the annotated word is correct by clicking on "Accept". Alternatively, h/she can correct the input in the same manner as explained before. Verification, compared to annotating from scratch, seems to have a positive effect on the human effort required (in terms of time, at least), as well as on the accuracy of the human annotators – see 4 below.

The verified data serves next as training data for developing a subsequent classifier. However, all previously created and verified training sets (i.e. excluding the gold standard) are automatically merged to form one single training set. The training data for each subsequent classifier is thus incrementally more, with a predicted increase in accuracy.

This bootstrapping process is repeated until the annotator is satisfied with the results. Thereafter, the project is handed over to the system developer, who can then continue to experiment with feature selection options and parameter optimization, in order to create the classifier that will deliver best results.

A few last notes on the system: *TurboAnnotate* runs under the Linux environment and requires Perl 5.8 and Tk. TiMBL (version 5.1) also needs to be installed, and the bin directory needs to be added to the PATH environment variable. Source code is freely available under an open-source license at http://www.nwu.ac.za/ctext.

4. Results

We evaluate our project in terms of two general criteria, viz. accuracy, and human effort (in terms of time). All evaluations are done on Afrikaans and Setswana, involving four human annotators in total; two of the annotators are well-experienced in annotating, while the other two could be considered novices in the field. Our results and main findings are discussed in the following subsections.

4.1. Accuracy

For evaluating our effort, we are interested in two kinds of accuracy: (1) classifier accuracy (i.e. how accurate is a classifier, and how much data is needed to approach the gold standard?), and (2) human accuracy (i.e. does *TurboAnnotate* contribute towards more accurate annotations by human annotators?). Accuracy is expressed as a percentage of correctly annotated words (not instances/hyphens) over the total number of words (either the gold standard, or another portion of data, depending on the evaluation). In all evaluations of classifiers, the gold standard was excluded as training data.

Table 1 clearly shows a huge difference between Afrikaans and Setswana: with only 200 training words, the Setswana classifier already reaches an accuracy of 94.50%, while the Afrikaans classifier only reaches 38.60%. This can be ascribed to the highly systematic CV-syllable structure of Setswana, thus proving that hyphenation for Setswana is rather trivial (a rule-based hyphenator with even less human effort would probably result in even better accuracy scores). For Afrikaans, however, the improvement is significant: after only 2,000 annotated words, a human annotator would have to correct only 3 out of every 10 words, thus promising less human effort towards the development of a state-of-the-art hyphenator (even more so when larger training sets will be used).

Training Data	Accuracy: Afrikaans	Accuracy: Setswana
200	38.60%	94.50%
600	54.00%	98.30%
1000	58.30%	98.80%
2000	68.50%	98.90%

Table 1: Classifier accuracy

To determine human accuracy, we created two previously unseen datasets of 200 words each for each language. The first dataset was annotated by each annotator in an ordinary text editor, by inserting a chosen symbol between syllables. The second dataset was annotated using *TurboAnnotate*. The mean accuracy of the annotators on the first dataset was 93.25%, while on the second dataset 98.34% - a difference of more than 5% (see Table 2). While this may seem insignificant, it could have an important impact when annotating large datasets for machine learning purposes.

Annotation Tool	Accuracy	Time (s)
Text Editor (200 words)	93.25%	1325
TurboAnnotate (200 words)	98.34%	1258

Table 2: Comparison of human accuracy and effort

It thus proves that *TurboAnnotate* could not only ensure higher accuracy in human annotations, but could also save on human effort required (at least in the case of Afrikaans). The latter aspect is evaluated in more depth in the next section.

4.2. Human effort

To determine whether *TurboAnnotate* would be beneficial in terms of time saved on annotation, we asked two questions: (1) Is it faster to annotate with *TurboAnnotate*, than with a text editor?; and if so, (2) What would be the predicted saving on human effort on a large dataset?

From Table 2 it is evident that it is more than 1 minute faster to annotate 200 words with *TurboAnnotate*, than with a text editor. On a larger dataset, say of 40,000 words, that would mean a small difference of only circa 3.5 uninterrupted human hours.

However, if one takes the effect of bootstrapping into consideration, a different picture arises. In the fourth iteration of the bootstrapping process (i.e. after the classifier was trained with 600 words), annotation time was measured, as well as after the last iteration (i.e. after the classifier was trained with 2,000 words). Annotators were not made aware of these time measurements, thus not influencing their focus on accuracy. Table 3 represents the mean results, indicating that the bootstrapping process has a significant positive impact on human effort.

Number of Words in Training Set	Time (s) to Annotate 200 words
0	1258
600	663
2000	573

Table 3: Human effort using *TurboAnnotate*

If we extrapolate this to a dataset of 40,000 words, and assume that the accuracy of the classifier does not improve (i.e. no further bootstrapping), then using *TurboAnnotate* could save almost 42 uninterrupted human hours – a saving of almost 57%.

5. Conclusions

In this paper, we have presented the design and functionality of our free and open-source tool, *TurboAnnotate*, which is used for creating gold standards and annotating lexical data through bootstrapping.

Future work includes extending the tool's capability to be scaled easily for other lexical annotation tasks, such as creating lexicons for spelling checkers (by using n-grams instead of machine learning to predict whether a word is correctly spelled), and creating data for various forms of morphological analysis (such as compound analysis, stemming, lemmatization, etc.). We will also focus on further improving the GUI, as well as a network solution for multi-users. Instead of TiMBL, we will also experiment with C5.0, since commercial licenses for the latter is rather affordable, an aspect that is relevant for the commercial exploitation of HLTs, even more so in developing countries.

6. Acknowledgements

This work was supported by a grant from the South African National Research Foundation (GUN: FA2004042900059). We also acknowledge the inputs and contributions of Ansu Berg, Handré Groenewald, Pieter Nortjé, Suléne Pilon, Rigardt Pretorius, Martin Schlemmer, and Wikus Slabbert.

7. References

[1] E. Garrett, "2nd Call for papers: Resource-scarce language engineering at ESSLLI 2006", Feb 20, 2006, http://pvs.csl.sri.com/mail-archive/pvs/msg02605.html.

[2] A. Eisele and D. Ziegler-Eisele, "Towards a road map on human language technology: natural language processing", [Online document], Mar 2002, [cited Mar 20, 2007], Available HTTP: http://www.elsnet.org/dox/rm-eisele-v2.pdf.

[3] M. Davel and M. Peche, "DictionaryMaker User Manual, Version 2.0(i)", Sep 14, 2006, http://dictionarymaker.sourceforge.net/.

[4] "Alchemist v2.0: A GUI-based tool for analyzing morphemes and creating morphological gold-standards in XML format", [Online document], [cited Apr 26, 2004], Available HTTP: http://linguistica.uchicago.edu/alchemistv2_0_final.pdf.

[5] M. Davel and E. Barnard, "A default-and-refinement approach to pronunciation prediction", Proc. of the Symposium of the Pattern Recognition Assoc. of South Africa, South Africa, November 2004, pp. 119-123.

[6] M. Davel and E. Barnard, "Bootstrapping for language resource generation", Proc. of the Symposium of the Pattern Recognition Assoc. of South Africa, South Africa, November 2003, pp. 97–100.

[7] C. Sprague, "Creating Morphological Descriptions with Alchemist", [Online document], [cited Jul 20, 2006], Available HTTP: http://linguistica.uchicago.edu/Sprague_2006.pdf.

[8] W. Daelemans, A. Van den Bosch, J. Zavrel and K. Van der Sloot, "TiMBL: Tilburg Memory Based Learner, Version 5.1, Reference Guide", ILK Technical Report 04-02, 2004.