AfriBooms: An Online Treebank for Afrikaans

Liesbeth Augustinus[*], Peter Dirix[*], Daniel van Niekerk[†], Ineke Schuurman[*], Vincent Vandeghinste[*], Frank Van Eynde[*], Gerhard van Huyssteen[†]

[*]Centre for Computational Linguistics
KU Leuven (University of Leuven), Leuven, Belgium
{firstname.lastname}@kuleuven.be
[†]Centre for Text Technology
North-West University, Potchefstroom, South Africa
{firstname.lastname}@nwu.ac.za

Abstract

Compared to well-resourced languages such as English and Dutch, natural language processing (NLP) tools for Afrikaans are still not abundant. In the context of the AfriBooms project, KU Leuven and the North-West University collaborated to develop a first, small treebank, a dependency parser, and an easy to use online linguistic search engine for Afrikaans for use by researchers and students in the humanities and social sciences. The search tool is based on a similar development for Dutch, i.e. GrETEL, a user-friendly search engine which allows users to query a treebank by means of a natural language example instead of a formal search instruction.

Keywords: Afrikaans, treebank, dependency parser, corpus search tool

1. Introduction

Afrikaans is a West Germanic language spoken as a first language by about 7 million people in South Africa and Namibia and by many millions more as a second language. It can be considered a daughter language of Dutch, as it originates in 17th-century Dutch dialects, brought to southern Africa by settlers from the Netherlands. Although there are some influences from Malay, Portuguese, Bantu, and Khoisan languages, Dutch and Afrikaans are still more or less mutually comprehensible. One of the main distinctions between Afrikaans and Dutch is a simplification of Dutch morphology in Afrikaans, e.g. dropping the nominal gender distinction and only keeping two verb forms for all but the most common verbs (present/infinitive and past participle).

In recent years, several NLP tools have been developed for Afrikaans, cf. Grover et al. (2011) for an overview of the available tools. Compared to well-resourced languages such as English and Dutch, however, there are fewer well-performing tools available for Afrikaans.

In this paper, we describe the development of an Afrikaans treebank (Section 2.), a dependency parser (Section 3.), and a tool for querying the treebank (Section 4.).

2. Treebank

The Afrikaans part of the NCHLT[1] Annotated Text Corpora (Puttkammer et al., 2014) was used as a basis for the development of the treebank. This corpus was selected because it had already been annotated with part-of-speech (POS) tags and word lemmas in a previous project (Eiselen and Puttkammer, 2014). The scope of work described below thus involved additional annotation for word dependencies within sentence context given the pre-existing POS tags and lemmas.

[1]South African National Centre for Human Language Technologies

2.1. Development

The NCHLT text corpus, consisting mainly of government domain documents, contains a number of incomplete sentence fragments which could not be annotated sensibly for dependency structure. The first action performed was filtering the corpus so that only complete sentences were annotated. The POS annotation in the NCHLT corpus was based on a fine-grained tag set developed specifically for Afrikaans (Pilon, 2005). This included labelling punctuation tokens as either “sentence-medial” or “sentence-final”. Filtering the corpus for suitable sentences thus involved keeping all entries that start with a capitalised token, end with “sentence-final” punctuation and contain at least one token considered to be a verb. Other entries were simply discarded; the resulting corpus statistics can be seen in Table 1.

Subset	Tokens/Words	Lines/Sentences
NCHLT corpus before filtering		
training set	55386	2610
test set	5834	329
NCHLT corpus after filtering		
training set	43895	1663
test set	5381	271

Table 1: NCHLT corpus division before and after filtering for valid sentences.

As some of the information in the original POS tag set may be superfluous for determining the sentence dependency structure and the additional granularity increased the difficulty and hence the reliability of the manual part of the annotation task, the POS tag set was simplified to a largely universal set of POS tags (Petrov et al., 2012). Table 2 contains the resulting tag set used with the number of original tags that was mapped to each simplified tag.

For the dependency relation annotation, a subset of the

POS tag	Afr. tags	Description
ADJ	6	Adjectives
ADP	1	Prepositions
ADV	19	Adverbs
CONJ	2	Conjunctions
DET	2	Determiners
NOUN	19	Nouns (including proper nouns)
NUM	13	Numerals
PRON	32	Pronouns
PRT	9	Particles
PUNCT	4	Punctuation
VERB	15	Verbs (including auxiliary verbs)
X	24	Catch-all class (including abbreviations and interjections amongst others)

Table 2: The POS tag set used with the number of sub-classes mapped from the original Afrikaans tag set used in the NCHLT corpus.

Stanford tag set was adopted and the conventions of de Marneffe et al. (2006) and de Marneffe and Manning (2008) were applied. This was done by presenting the full set of tags and hierarchy to annotators in the annotation protocol (see Table 3), but allowing them to fall back onto more generic tags where they were unsure about specific relations. This resulted in a subset of the tags being applied in practice, see Section 2.2. for a further discussion of this.
To bootstrap the annotation of the corpus, a preliminary parser was constructed using the rule-based A2DC[2] Afrikaans-to-Dutch converter (Pilon et al., 2010) and the Frog[3] dependency parser for Dutch (van den Bosch et al., 2007). Word dependency relations and tags were combined with the pre-existing POS tags, mapped to the adopted tag sets and converted to the annotation tool file format for manual correction by the annotators. The annotation tool used during the manual correction was the BRAT[4] web-based rapid annotation tool (Stenetorp et al., 2012) of which the standard dependency annotation configuration was customised for this task.
Informal evaluations during development showed that only about 20 to 30% of the connections from the preliminary (A2DC+Frog) parser were kept during manual annotation. One possible reason for the low level of success is different conventions in terms of dependency structure followed by Frog compared to the Stanford protocol. One of the differences in convention in the Stanford protocol is that *content* words are related to each other, where possible, directly instead of indirectly. For example, content words such as verbs and nouns are typically directly connected using the `dep:conj` tag with coordinating conjunctions being dependent on these content words via the `dep:cc` tag, whereas the output that was obtained from Frog typically represented the dependency *through* the coordinating conjunction. Consequently, only around 10% of the corpus was annotated manually from the initial A2DC+Frog parse. This set of sentences was used to improve the parse for subsequent manual annotation by training a neural network-based parser (Titov and Henderson, 2007), cf. section 3. During the process of annotation two more iterations of updating and applying the parser were performed to incrementally improve the starting parse (and therefore the time spent correcting annotations). The parser is discussed in section 3.
As a final post-processing step, all sentences were validated for adherence to the fundamental constraints specified in the annotation protocol, specifically:

- *Graph completeness*: Each sentence must form a single complete graph, i.e. all words must be reachable from the root node.
- *Dependence restriction*: Words may have multiple dependants but not multiple heads.
- *Projectivity*: Connection lines between words should not cross each other.

2.2. Results

The corpus was annotated by one primary annotator and a subset containing 943 words was annotated by a second annotator with experience in Afrikaans linguistics. For this subset we calculated the inter-annotator agreement (IAA) both in terms of the *labelled attachment score* (LAS) and the *unlabelled attachment score* (UAS) averaged over words (Nivre et al., 2007). The UAS was 88.9% while the LAS was 82.5%. This indicates that the task defined in the annotation protocol was sufficient to result in consistent annotation. Further analysis of the annotator differences pointed out that the primary annotator often used more generic tags where more specific ones were appropriate. Out of the 31 tags defined in the annotation protocol, only 20 were used in the final corpus and only 18 in significant numbers. Table 3 contains the number of instances of each tag in the corpus. Here we can see that the annotator made extensive use of relatively generic tags such `dep`, `dep:mod` in the case of modifiers and `dep:arg:comp:obj` in the case of objects (with no references to indirect objects).

[2]`http://sourceforge.net/projects/d2ac-a2dc/`
[3]`http://ilk.uvt.nl/frog/`
[4]`http://brat.nlplab.org`

Dependency tag	Freq.	Description
dep	1009	dependent
dep:**punct**	4497	punctuation
dep:**root**	1870	root
dep:**aux**	2534	auxiliary (verb)
dep:**conj**	2359	conjunct
dep:**cc**	1886	coordination (e.g. to conjunctions)
dep:**arg**	1130	argument
dep:arg:**subj**	2605	subject
dep:arg:**comp**	3	complement
dep:arg:comp:**obj**	3763	object
dep:arg:comp:obj:**dobj**	111	direct object
dep:arg:comp:obj:**iobj**	0	indirect object
dep:arg:comp:obj:**pobj**	6106	object of preposition
dep:arg:comp:**compl**	0	complementiser
dep:arg:comp:**mark**	5	marker (introducing adverbial clause)
dep:arg:comp:**rel**	0	relative (introducing relative clause)
dep:arg:comp:**acomp**	0	adjectival complement
dep:**mod**	11120	modifier
dep:mod:**advcl**	0	adverbial clause modifier
dep:mod:**tmod**	0	temporal modifier
dep:mod:**amod**	2447	adjectival modifier
dep:mod:**num**	462	numeric modifier
dep:mod:**number**	0	element of compound number
dep:mod:**appos**	0	appositional modifier
dep:mod:**abbrev**	63	abbreviation modifier
dep:mod:**adv**	0	adverbial modifier
dep:mod:adv:**neg**	0	negation modifier
dep:mod:**poss**	830	possession modifier
dep:mod:**prt**	1375	phrasal verb particle
dep:mod:**det**	5101	determiner
dep:mod:**prep**	0	prepositional modifier

Table 3: The Stanford dependency tag set with number of occurrences in the corpus.

The above evaluation, as well as the evaluation of the resulting parser in the following section, suggests a consistently annotated initial corpus. However, there remains considerable room for improvement in terms of finer-grained linguistic detail. The final corpus is released in two parts (the training set and the test set), with accompanying lemmas and POS tags in XML format,[5] and is available online from the South African Language Resource Management Agency (RMA).[6]

3. Dependency parser

3.1. Development

Initially, we envisaged a parser based on A2DC and Frog. Such a system would in principle be able to reuse a POS tagger, lemmatizer and dependency parser for Dutch. While this approach was valuable for bootstrapping the annotation process, the accuracy obtained was limited in practice. Two factors likely contributing to the limited accuracy were:

- The cumulative effect of the propagation of errors resulting from a sequence of imperfect components, especially the word-based conversion from Afrikaans to Dutch which does not at this stage take into account POS.
- The differences in convention and mapping between the Stanford protocol and Frog.

The implemented parser consists of a tokenizer, a POS tagger, a lemmatizer and a dependency parsing component. The tokenizer simply splits on white-space and punctuation. The POS tagger and lemmatizer components were developed in a separate project (Eiselen and Puttkammer, 2014).[7] For the dependency parser, we followed the approached described in Titov and Henderson (2007), and specifically the IDP implementation.[8] We used the default parameters suggested in the documentation and with cut-off free parameters both set to 5.

3.2. Results

Where applicable, the parsers were trained on the training set. We used a 10% randomly selected development set during training and tested using the test set. The first line in the table shows the score for the parser component given known POS labels and lemmas and may be compared to the IAA numbers for manual annotation.

Parser	UAS	LAS
IDP parser component	0.922	0.901
composite parser	0.888	0.814
A2DC + Frog	0.432	0.218

Table 4: Results for different parser configurations.

As expected, the parser component scores better with known POS tags and lemmas, with only a slight drop in accuracy for the composite parser, possibly owing to the high accuracy achieved by the POS tagger. The composite parser is released as a deliverable for this project, trained on the entire corpus. The individual components will all be available from the RMA.[9]

4. Search tool: GrETEL

GrETEL (**Gr**eedy **E**xtraction of **T**rees for **E**mpirical **L**inguistics) was originally developed to facilitate searching in Dutch treebanks (Augustinus et al., 2012; Augustinus et al., 2013).[10] In this project, it was adapted to be

[5]Specifically the FoLiA XML as used by the Frog parser; see `http://ilk.uvt.nl/folia`.

[6]See: `http://rma.nwu.ac.za/index.php/resource-catalogue/afribooms.html`

[7]See: `http://rma.nwu.ac.za/index.php/afrikaans-nchlt-lemmatiser.html`

[8]Freely available under GNU GPL version 3 at `http://cui.unige.ch/~titov/idp/`.

[9]`http://rma.nwu.ac.za`

[10]`http://gretel.ccl.kuleuven.be`

used for the Afrikaans treebank annotation scheme. *GrETEL4Afrikaans* is freely available through the GrETEL website.[11]
There are two ways to query a treebank using GrETEL: by means of a natural language example (section 4.1.) or by means of an XPath query (section 4.2.).

4.1. Example-based querying

The example-based querying approach is a query procedure in six steps.

1. Example The user enters an example of the construction (s)he is interested in. For instance, if one wants to look up progressive constructions consisting of a form of *wees* 'be', *besig om te* 'busy for to', and a verb, one can use the example in (1) as a starting point for querying the treebank.

(1)	*Hy*	*is*	*besig*	*om*	*te*	*lees.*
	he	is	busy	for	to	read
	'He is reading.'					

2. Parse GrETEL automatically parses this example and returns the result to the user. If the parse is correct, the user can continue to the next step. If the annotation does not make sense, the user is advised to choose another input example.

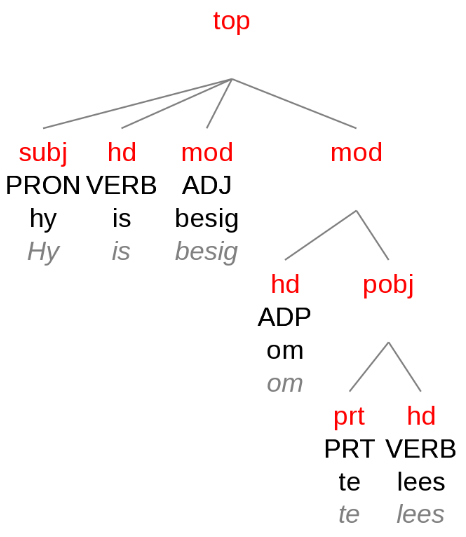
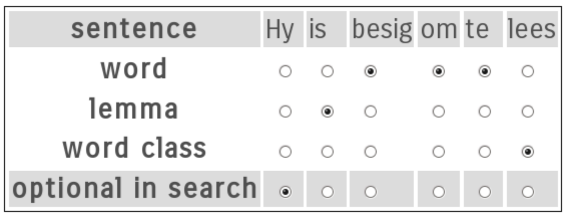


Figure 1: Parse of the input construction

3. Selection matrix GrETEL presents the input example in a matrix, cf. Figure 2.
In this step the user can choose which parts of the construction are relevant for the construction under investigation. For each word, the user can indicate whether the word class (POS), the lemma, or the exact word form should be included in the search instruction. In this case, *lemma* is indicated for *is*, *word* is indicated for *besig*, *om*, and *te*, and word class is indicated for *lees*, as it does not matter with which verb the construction is combined. The subject of the sentence is not relevant for the search construction, but is needed in the input in order to get a valid parse. In the matrix, the subject is indicated as *optional in search*.



sentence	Hy	is	besig	om	te	lees
word	○	○	●	●	●	○
lemma	○	●	○	○	○	○
word class	○	○	○	○	○	●
optional in search	●	○	○	○	○	○

Figure 2: Selection matrix

4. Treebank selection In this step the user can select the treebank that (s)he would like to query. Currently the NCHLT treebank is the only option in *GrETEL4Afrikaans*, but in future work other treebanks may be included as well (cf. section 5.).

5. Query Based on the information provided in the selection matrix, GrETEL extracts a subtree from the parse tree, cf. Figure 3. In addition to the lexical information indicated in the selection matrix, the dependency relations of the relevant nodes are included in this *query tree* as well.

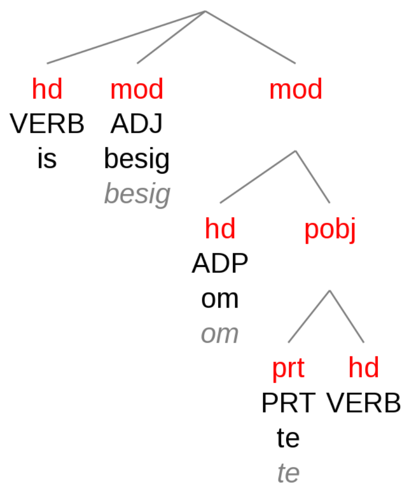


Figure 3: Query tree based on the input example

GrETEL automatically converts the query tree into an XPath expression,[12] which is used for the actual treebank search. The query generated from the query tree in Figure 3 is given in (2).

(2)
```
//node[node[@rel="hd" and
@pt="VERB" and @lemma="is"] and
node[@rel="mod" and @pt="ADJ" and
@word="besig" and @lemma="besig"]
and node[@rel="mod" and
node[@rel="hd" and @pt="ADP"
and @word="om" and @lemma="om"]
and node[@rel="pobj" and
```

[11]http://gretel.ccl.kuleuven.be/afribooms

[12]http://www.w3.org/TR/xpath

```
node[@rel="prt" and @pt="PRT" and
@word="te" and @lemma="te"] and
node[@rel="hd" and @pt="VERB"]]]]
```

In the basic search mode, the query is not presented to the user at this stage. In the advanced search mode, users can adapt the XPath expression in order to refine or generalize the search instruction.

6. Results If there are constructions in the treebank matching the XPath expression, GrETEL presents the results as a list of sentences, with the matching part emphasized. The user can click on any of these sentences in order to visualize the constructions as syntax trees. GrETEL finds 6 results for the query in (2). Some examples are given in (3–4). The results show the *greedy* nature of GrETEL: It not only returns constructions in which the items defined in the search are adjacent, but also returns examples in which those elements are discontinuous. Such constructions would be harder to identify in a *flat* corpus.

(3) ten opsigte van voorsiening **is** Eskom druk **besig om** die bekendstelling van mede-opwekkingsprojekte as ‘n noodsaaklikheid **te verseker**. [NCHLT.p.1138.s.1138]

(4) ons **is** hard **besig om** die goedkeuring en oprigting van gasturbineprojekte **te bespoedig**. [NCHLT.p.1140.s.1140]

The advantage of example-based querying is that the user does not need to be familiar with XPath, nor with the underlying XML structure of the trees, nor with the detailed grammar implementation that is used by the parser or the treebank annotators.

4.2. XPath search

Although example-based querying has many advantages, querying a treebank using XPath directly enhances the query flexibility compared to the example-based approach. Therefore, the second way of querying the corpora in GrETEL consists of directly formulating an XPath query describing the syntactic pattern the user is looking for. This query is then processed in the same way as the automatically generated query in the first approach.

XPath querying allows more flexibility in the type of patterns that are searched. As mentioned in section 4.1., one can manually adapt the generated XPath query before querying the treebank in the advanced search mode of example-based querying. This can be seen as an intermediate approach, as XPath is easier to understand and adapt than to construct from scratch.

5. Conclusions and future work

We have described the development of an Afrikaans treebank and parser, as well as the inclusion of those resources into the search engine GrETEL.

Currently *GrETEL4Afrikaans* only contains the NCHLT treebank, which is rather modest in size, but the search engine is organised in such a way that a larger treebank can be included, e.g. a syntactically annotated version of the Taalkommissie corpus (CTexT, 2011). We intend to make a parsed version of this corpus available for research purposes. It could certainly be used for corpus analysis in Afrikaans, as was already done for the IPP effect using an earlier in-house version of the search tool (Augustinus and Dirix, 2013).

6. Acknowledgements

The AfriBooms project was sponsored by the Dutch Language Union and the Department of Arts and Culture of the Government of South Africa in order to create a manually checked Afrikaans treebank which could then be used for bootstrapping other tools and to support corpus research for Afrikaans.

7. Bibliographical References

Augustinus, L. and Dirix, P. (2013). The IPP effect in Afrikaans: A Corpus Analysis. In *Proceedings of the 19th Nordic Conference of Computational Linguistics (NODALIDA 2013) - NEALT Proceedings Series 16*, pages 213–225, Oslo.

Augustinus, L., Vandeghinste, V., and Van Eynde, F. (2012). Example-Based Treebank Querying. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*, pages 3161–3167, Istanbul.

Augustinus, L., Vandeghinste, V., Schuurman, I., and Van Eynde, F. (2013). Example-Based Treebank Querying with GrETEL – now also for Spoken Dutch. In *Proceedings of the 19th Nordic Conference of Computational Linguistics (NODALIDA 2013) - NEALT Proceedings Series 16*, pages 423–428, Oslo.

de Marneffe, M.-C. and Manning, C. D. (2008). The Stanford typed dependencies representation. In *Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation at COLING 2008*, pages 1–8, Manchester, UK.

de Marneffe, M.-C., MacCartney, B., and Manning, C. D. (2006). Generating Typed Dependency Parses from Phrase Structure Parses. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2012)*, pages 449–454, Genoa.

Eiselen, R. and Puttkammer, M. J. (2014). Developing Text Resources for Ten South African Languages. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC 2014)*, pages 3698–3703, Reykjavik.

Grover, A. S., van Huyssteen, G. B., and Pretorius, M. W. (2011). A Technology Audit: The State of Human Language Technologies (HLT) R&D in South Africa. In *Proceedings of PICMET'11: Technology Management In The Energy-Smart World (PICMET)*, pages 1693–1706, Portland, Oregon.

Nivre, J., Hall, J., Kübler, S., McDonald, R., Nilsson, J., Riedel, S., and Yuret, D. (2007). The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL shared task session of EMNLP-CoNLL*, pages 915–932.

Petrov, S., Das, D., and McDonald, R. (2012). A Universal Part-of-Speech Tagset. In *Proceedings of the 8th Inter-*

national Conference on Language Resources and Evaluation (LREC 2012), pages 2089–2096, Istanbul.

Pilon, S., van Huyssteen, G., and Augustinus, L. (2010). Converting Afrikaans to Dutch for technology recycling. In *Proceedings of the 2010 Conference of the Pattern Recognition Association of South Africa (PRASA-2010)*, pages 219–224, Stellenbosch.

Pilon, S. (2005). Outomatiese Afrikaanse woordsoortetikettering. Master's thesis, North-West University, Potchefstroom.

Stenetorp, P., Pyysalo, S., Topić, G., Ohta, T., Ananiadou, S., and Tsujii, J. (2012). BRAT: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107, Avignon, France.

Titov, I. and Henderson, J. (2007). Fast and Robust Multilingual Dependency Parsing with a Generative Latent Variable Model. In *Proceedings of the Joint Meeting of the Conference on Empirical Methods in Natural Language Processing and the Conference on Computational Natural Language Learning (EMNLP-CoNLL 2007)*, pages 947–951, Prague.

van den Bosch, A., Busser, B., Canisius, S., and Daelemans, W. (2007). An efficient memory-based morphosyntactic tagger and parser for Dutch. In *Computational Linguistics in the Netherlands: Selected papers from the Seventeenth CLIN Meeting*, pages 191–206, LOT, Utrecht.

8. Language Resource References

CTexT. (2011). *Taalkommissiekorpus*. CTexT North-West University, Potchefstroom, Version 1.1.

Puttkammer, M. and Schlemmer, M. and Bekker, R. (2014). *Afrikaans NCHLT Annotated Text Corpora*. South African Language Resource Management Agency, Potchefstroom, 1.0, ISLRN 139-586-400-050-9.