Example-Based Treebank Querying

Liesbeth Augustinus, Vincent Vandeghinste, Frank Van Eynde

Centre for Computational Linguistics
KU Leuven
liesbeth@ccl.kuleuven.be, vincent@ccl.kuleuven.be, frank@ccl.kuleuven.be

Abstract

The recent construction of large linguistic treebanks for spoken and written Dutch (e.g. CGN, LASSY, Alpino) has created new and exciting opportunities for the empirical investigation of Dutch syntax and semantics. However, the exploitation of those treebanks requires knowledge of specific data structures and query languages such as XPath. Linguists who are unfamiliar with formal languages are often reluctant towards learning such a language. In order to make treebank querying more attractive for non-technical users we developed GrETEL (Greedy Extraction of Trees for Empirical Linguistics), a query engine in which linguists can use natural language examples as a starting point for searching the Lassy treebank without knowledge about tree representations nor formal query languages. By allowing linguists to search for similar constructions as the example they provide, we hope to bridge the gap between traditional and computational linguistics. Two case studies are conducted to provide a concrete demonstration of the tool. The architecture of the tool is optimised for searching the LASSY treebank, but the approach can be adapted to other treebank lay-outs.

Keywords: Dutch, treebank, querying

1. Introduction

The recent construction of large linguistic treebanks (or syntactically annotated text corpora) for spoken and written Dutch such as CGN (van der Wouden et al., 2003), (Van Eynde, 2009), LASSY (van Noord et al., 2006; van Noord et al., in press), and Alpino Treebank (van der Beek et al., 2002) has created new and exciting opportunities for the empirical investigation of Dutch syntax and semantics. However, the exploitation of those treebanks usually requires knowledge of specific data structures and/or query languages, which may discourage linguists to use them.

In this paper, we present a user-friendly search application for the exploitation of treebanks by linguists who are not familiar with nor interested in data formats or query languages. By allowing linguists to search for similar constructions as the example they provide, we hope to bridge the gap between traditional linguistics and treebank builders. This conforms with one of the project goals of CLARIN[1] to open up language resources for human and social sciences. In section 2 we will shortly discuss the most common querying issues. In section 3 we will present the concept of example-based querying and the architecture of our search application, which will be illustrated by the elaboration of two examples in section 4. Finally, we will draw conclusions and touch on some topics for future research in section 5.

2. Querying Issues

In the literature on treebank querying we are faced with the same problems over and over again. A major obstacle is the *limited user-friendliness* of the query languages and search tools. That problem is closely related to another issue of treebank mining: the *lack of standardisation* in both treebanks and query languages. See amongst others: Lai and Bird (2004), Hellmann et al. (2010), and Štepánek and Pajas (2010).

Nowadays there are many natural language treebanks and almost as many formal languages to query those treebanks. For example, the Penn Treebank (Marcus et al., 1993) should be queried with TGrep2 (Rohde, 2005), the TIGER Treebank (Brants et al., 2002) and CGN can be queried with TIGERSearch (Lezius, 2002), and for LASSY the W3C standards XPath[2] and XQuery[3] can be used for searching and extracting information from the treebank with applications like dtsearch (Bouma and Kloosterman, 2002; Bouma and Kloosterman, 2007) or DACT.[4] Because of that *overload of query languages, annotation formats and data structures*, many linguists give up treebank mining as they do not easily find what they are looking for. It requires time and effort to learn a formal language, since queries get long and complex relatively quickly.

As it is quite a hassle to learn such query languages and data structures, some search tools offer a GUI in order to shield the linguist from the internal formalisation of the treebank, e.g. TIGERin (Voormann and Lezius, 2002), and SearchTree (Nygaard and Johannesen, 2004). Unfortunately, the linguist still has to be familiar with the exact tree lay-out and hence the underlying linguistic theories of the treebank builders in order to formulate what (s)he is looking for. Therefore such GUIs are in fact *less user-friendly* than they are supposed to be.

Since *standardisation*[5] in the highly evolving field of treebank building and querying is still far off,[6] we decided

[1] http://www.clarin.eu

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

[3] http://www.w3.org/TR/xquery

[4] http://rug-compling.github.com/dact/

[5] Although a standard format for linguistic annotation is not defined yet, the FoLiA format (van Gompel, 2011) is a first attempt towards such a format for Dutch.

[6] There are some ongoing efforts towards standardisation: W3C standard technologies are commonly used for natural lan-

to tackle the problem in another way. Instead of developing yet another query language or designing yet another GUI, we present a query engine which does not ask for any formal input query. As input, the tool takes something linguists are familiar with: *natural language*.

3. Example-Based Querying

Since linguists tend to start their research from example sentences, example-based querying allows to use those examples as a starting point for treebank search. Work related to our approach are the Linguist's Search Engine (Resnik and Elkiss, 2005), a tool that makes use of example-based querying, and the TIGER Corpus Navigator (Hellmann et al., 2010), which is a Semantic Web system used to classify and retrieve sentences from the TIGER corpus on the basis of abstract linguistic concepts.

We present GrETEL (**Gr**eedy **E**xtraction of **T**rees for **E**mpirical **L**inguistics), a tool for example-based querying. The implementation is optimised for querying LASSY Small,[7] the manually annotated part of the LASSY treebank (van Noord et al., 2006; van Noord et al., in press), which consists of unordered dependency trees. Since those trees are in XML format, they can be queried with XPath and XQuery.

Figure 1 presents the architecture of GrETEL. The system takes as input an example of the syntactic construction the user is looking for. The input construction is not necessarily a full sentence. The user can indicate which parts of the input construction are relevant. Next, the construction is parsed with the Alpino parser (van Noord, 2006) and the relevant parts are annotated in the parse tree. Then, the *Subtree Finder* looks for the annotations in the parse tree in order to 'cut out' the subtree. That subtree is the input of the *XPath Generator*, which converts the subtree into an XPath query. After the conversion the user has the option to adapt the query if necessary. In the final step the query is matched against the LASSY Small treebank. If any matching constructions are found, the results are displayed to the user.

Our approach of example-based querying could be adapted to work for any other treebank if there exists a parser which outputs trees similar to the trees in the treebank, such as the Charniak parser (Charniak, 1997) and the Penn Treebank. Furthermore, terminal nodes of the parse trees should contain part of speech (POS), lemma, and token (word form). In order to be compatible with GrETEL, the treebank should be converted to the Alpino XML structure or a similar XML data structure. Such a conversion is generally possible for treebanks with a tree-like data structure, e.g. Penn Treebank, and the monolingual sides of the treebanks described in (Kotzé et al., 2012). Less important aspects are the linguistic framework of the treebank (e.g. whether the treebank is phrase structure-based or dependency-based) and the (natural) language of the treebank.

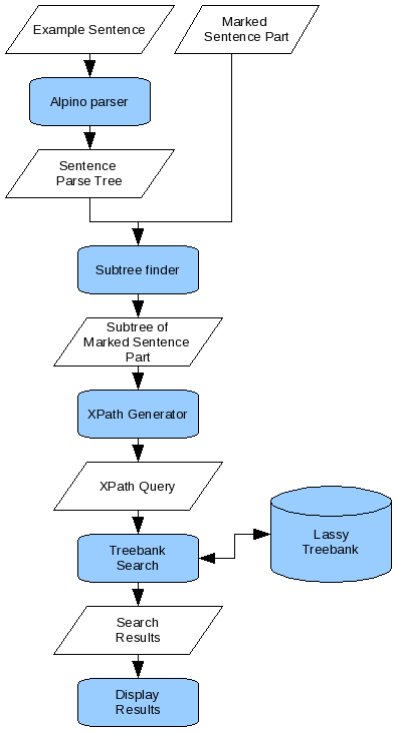

Figure 1: Architecture of GrETEL

4. Two Case Studies

Two examples are presented in order to give a detailed elaboration of treebank mining using GrETEL. The first case considers Dutch nominalisations (section 4.1). The second example investigates the position of separable verb particles in Dutch subclauses (section 4.2), which is a more complex case as the word order is taken into account. Both examples will show that GrETEL returns sentences similar to the input example.

4.1. Case 1: Nominalisations

If one is interested in deverbal nominalisations with determiner *het* [E: 'the'] and a direct object introduced by preposition *van* [E: 'of'], a possible input sentence is (1).

(1) **Het doden van olifanten** is verboden.
the kill of elephants is forbidden
'Killing elephants is prohibited.'

guage treebanks: XML for data storage, XPath/XQuery for searching treebanks (e.g. LASSY, Alpino Treebank). Sometimes tools for linguistic purposes are extensions of those fundamental standards. An example is LPath (Lai and Bird, 2010), a query language based on XPath. Although it is easier to learn LPath if one is already familiar with XPath, such extensions to existing standards do not solve the problem of query language overload. A different approach to solve the standardization problem is the creation of a query language that covers several treebanks such as PML (Štepánek and Pajas, 2010).

[7]We have made some changes to the Lassy treebank and the Alpino parses of the input sentences, aiming at a generalisation in query patterns. The Alpino parser and the Lassy treebank do not allow unary branching. When a single noun occurs, it will hence not be placed under a *noun phrase* (NP). We have applied a transduction on the whole treebank to put such *bare* nouns and names under an NP, allowing unary branching in this case.

The input construction is presented in a matrix (Figure 2), allowing the user to indicate the parts of the sentence that are relevant for the syntactic construction (s)he is querying. Initially the *not relevant* button is selected for all words. The user can indicate for each word whether (s)he is interested in the part of speech (POS), the lemma or the token. If the user is looking for non-lexical similarities, *pos* should be selected. The *lemma* button should be indicated to abstract over word forms, and the *token* button should be selected for retrieving specific word forms.[8] If the input contains words that are no part of the target construction, the *not relevant* button should remain selected.

sentence	Het	doden	van	olifanten	is	verboden
pos	○	◉	○	◉	○	○
lemma	◉	○	◉	○	○	○
token	○	○	○	○	○	○
not relevant	○	○	○	○	◉	◉

Ordering filter ☐

Submit

Figure 2: Input matrix for example (1)

It should be noted that the dependency relation and the POS of all relevant nodes are taken into account. For example, if the *lemma* button would be selected for *doden* [E: 'to kill'], sentences like (2) will match. Sentences in which *doden* [E: '(the) death'] is a plural noun, such as (3) will not match. The tool thus considers lemmas and tokens *in context* instead of mere string matching.

(2) Het doden van mensen is verboden.
the kill of humans is forbidden
'Killing humans is prohibited.'

(3) Het aantal doden van dat treinongeval stijgt.
the amount deaths of that train accident rises
'The death toll from that train accident is rising.'

In order to find more matches, we only indicated *pos* for both the nominalised verb and the noun in its PP sister. The lemmas of the determiner *het* and the preposition *van* are indicated, since those words are inherent parts of the nominalised constructions we are looking for.

The input construction is parsed with the Alpino parser (van Noord, 2006), which is also used as a starting point for the treebank annotations of the LASSY treebank.
The information provided in the input matrix is then added to the parse tree, allowing us to extract a subtree from the full parse tree, as shown in Figure 3.[9] The top of the subtree is the lowest node that dominates the relevant items.

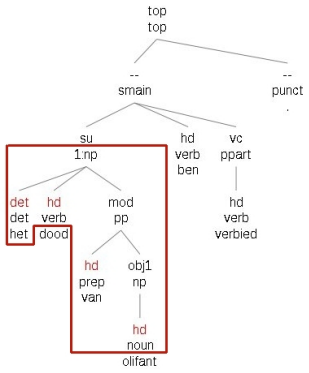

Figure 3: Alpino parse of example (1) with selected subtree

The subtree is used as input for the *XPath Generator*, which converts them into XPath queries. The query in (4) is generated from the subtree indicated in Figure 3.

(4)
```
//node[@cat="np" and node[@rel="det"
and @root="het" and @pos="det"] and
node[@rel="hd" and @pos="verb"] and
node[@rel="mod" and @cat="pp" and
node[@rel="hd" and @root="van" and
@pos="prep"] and node[@rel="obj1"
and @cat="np" and node[@rel="hd" and
@pos="noun"]]]]
```

The XPath query contains all elements that are present in the subtree (i.e. dependency relation `[@rel]`, POS `[@pos]` and for some nodes lemma `[@root]`).[10]
After the query is generated, the user has the option to choose between *basic* and *advanced* search mode.
In basic search mode, the XPath query is immediately matched against the corpus. The user has the option to search the complete LASSY Small treebank (65k sentences) or to select one or more subcorpora. For the query in (4) we have found 1288 hits in 1206 sentences in the complete corpus. Five matches are presented in (5 - 9).

(5) **Het samenleven van verschillende**
the together-live of different
bevolkingsgroepen gaat niet vanzelf .
communities goes not by-itself
'It is not easy for different communities to live together.'

(6) Controle op **het naleven van de regelgeving**
control on the comply of the rules
binnen de beschermde gebieden
within the protected areas
'Controlling the compliance of the rules within the protected areas'

[8] For invariable word forms such as *het* and *van* in example (1), it does not matter whether lemma or token is selected for those words.

[9] The graphical representation only shows per node (from top to bottom) the dependency relation, the syntactic category or POS, and the lemma.

[10] The dependency relation of the top node is omitted because it indicates a relation with the parent node, which is not included in the subtree.

(7) Satellieten hebben een erg brede kijk op het
satellites have a very wide view on the
aardoppervlak , en daarom worden zij ook
earth-surface , and therefore become they also
vaak gebruikt voor **het observeren van de**
often used for the observe of the
oceanen .
oceans
'Satellites have a very wide view on the surface of the earth, and therefore they are also often used for the observation of the oceans.'

(8) Hierin zoekt hij **het isoleren van de**
here-in looks for he the isolate of the
menselijke figuur , alweer in een indrukwekkend
human figure , again in an impressive
monumentaal gebeuren .
monumental happening
'In this he looks for the isolation of the human character, again in an impressive monumental event.'

(9) Door **het invullen van de enquête** kunt u dat
by the in-fill of the survey can you that
kenbaar maken .
knowable make
'You can indicate that by filling out the survey.'

The advanced search mode allows users who are familiar with XPath to adapt the query if necessary. One could for example cut off the last part of the query (i.e. the part of the query that corresponds to the head-noun[11] branch in the subtree), which results in (10).

(10)
```
//node[@cat="np" and node[@rel="det"
and @root="het" and @pos="det"] and
node[@rel="hd" and @pos="verb"] and
node[@rel="mod" and @cat="pp" and
node[@rel="hd" and @root="van" and
@pos="prep"] and node[@rel="obj1" and
@cat="np"]]]
```

Since the derived query (10) contains less constraints than the original one (4), matches found by (10) will include the matches found by (4). The results furthermore include (sub)trees with a name or a multiword expression as head of the PP. The abstract query finds 1391 hits in 1299, of which five examples are presented in sentences (11) - (14).

(11) Het bracht een positief advies uit voor
it brought a positive advice out for
goedkeuring van **het in de handel brengen van**
approval of the in the trade bring of
Humalog .
Humalog
'A positive advice was given for the approval of the trade with Humalog.'

(12) Maar de selectie van de gezichten voor de
but the selection of the faces for the
cover kwam Bono toe , en hij pleitte voor **het**
cover came Bono to , and he pleaded for the
opnemen van Bush .
up-take of Bush
'The selection of the cover face was up to Bono, and he argued for including Bush.'

(13) **Het ontstaan van het ABN** .
the originate of the ABN
'The emergence of ABN.'

(14) Hij groeide op in een woelig politiek
He grew up in a turbulent political
milieu , onder meer door **het uitbreken**
environment , amongst more by the outbreak
van de Tweede Wereldoorlog .
of the Second World War
'He grew up in a turbulent political situation, for example because of the outbreak of the Second World War.'

4.2. Case 2: Separable Verb Particles

Dutch verbs have a tendency to cluster. In the case of subclauses, it means that both the finite verb and all non-finite verbal elements form a verbal complex on the second pole. However, in some constructions non-verbal elements appear within the verbal complex (Haeseryn et al., 1997). A typical case of such constructions are sentences with separable verbs. Since the separable verb particle is closely related to the verb, it sometimes occurs within the verbal complex. An example of such a construction is given in (15), where *kennis* occurs between the verb forms *moet* and *maken* .

(15) De gedachte dat ze **moet kennis maken**
the thought that she **has acquintance make**
met haar schoonmoeder stemde haar niet
with her mother-in-law make-feel her not
gelukkig.
happy
'The thought that she had to meet her mother-in-law did not make her feel happy.'

Since we are only interested in the subclause, it is sufficient to present (16) to the system.

(16) dat ze moet kennis maken
that she has acquaintance make
'that she had to meet'

The input matrix of (16) is given in Figure 4. The example's parse and the extracted subtree are presented in Figure 5.

The XPath query generated from the subtree in Figure 5 is given in (17). It should be noted that the results do not only include sentences in which the separable verb particles interrupts the verbal complex, as word order is not taken into account. As examples in which the separable verb particle occurs out of the verbal complex are returned as similar

[11] The tag *noun* is only assigned to common nouns in the LASSY treebank. Proper nouns are indicated with the tag *name*.

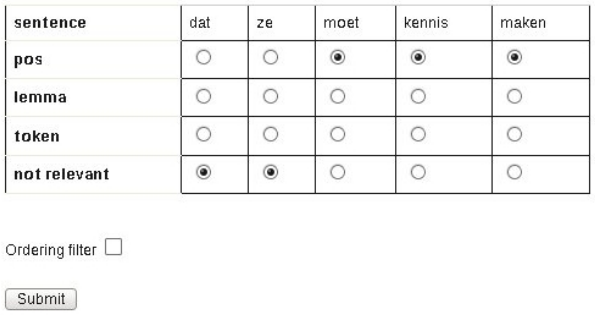

Figure 4: Input matrix for example (16)

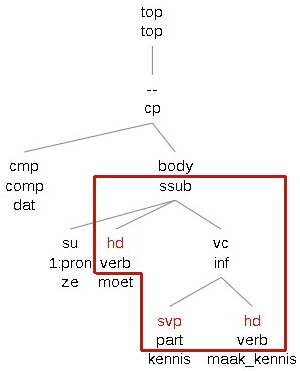

Figure 5: Alpino parse of example (16) with selected subtree

examples as well, the query finds the total amount of sentences with separable verb particles in subclauses: 98 hits in 95 sentences.

(17)
```
//node[@cat="ssub" and node[@rel="hd"
and @pos="verb"] and node[@rel="vc"
and @cat="inf" and node[@rel="svp"
and @pos="part"] and node[@rel="hd"
and @pos="verb"]]]
```

In order to investigate the position of separable verb particles with GrETEL we need (at least) two input sentences: one with a non-verbal element (separable verb particle) within the verbal complex and one in which the separable verb particle does not occur within the verb cluster. Example (15) is a possible input sentence to investigate the first scenario (interruption). A similar sentence without interruption is presented in (18); the subclause we have presented to the system is given in (19).

(18) De gedachte dat ze **kennis moet maken**
the thought that she **acquintance has make**
met haar schoonmoeder stemde haar niet
with her mother-in-law make-feel her not
gelukkig.
happy
'The thought that she had to meet her mother-in-law did not make her feel happy.'

(19) dat ze kennis moet maken
that she acquaintance has make
'that she had to meet'

Since the Alpino parser generates dependency trees, the structure of the parse of the non-interruption example will be exactly the same as the parse of the interruption example. The difference lies in the surface position of the terminal nodes, which is included in the `begin` feature of each node.[12] As the Alpino parser includes information on the position of terminal nodes, it is possible to respect the word order of the input sentence. However, the formulation of such XPath queries is rather complex, because the absolute word position values have to be compared in a relative way. In order to do that computation automatically, we have created an *Ordering Filter* that takes the surface order of the words into account while generating the XPath query.

The following query (20) is generated from the subtree derived from the interruption example *dat ze moet kennis maken*. The input matrix looks similar to Figure 4, but the *Ordering Filter* box is checked:

(20)
```
//node[@cat="ssub" and node[@rel="hd"
and @pos="verb" and @begin
< ../node[@rel="vc" and
@cat="inf"]/node[@rel="svp"
and @pos="part"]/@begin] and
node[@rel="vc" and @cat="inf" and
node[@rel="svp" and @pos="part"
and @begin < ../node[@rel="hd" and
@pos="verb"]/@begin]]].
```

That query matches with 4 hits in 4 sentences, which are presented in (21 - 24).

(21) Als we echt **willen vooruit komen** met Europa
If we really want forward come with Europe
moeten we geen "verdrag min" maar een
must we no "treaty minus" but a
"verdrag plus" afsluiten .
"treaty plus" off-close
'If we really want to move forward with Europe we should not make a "treaty minus" but a "treaty plus".'

(22) Dit is het schooljaar vóór uw kind in een
this is the school-year before your child in a
welbepaalde school effectief **zal school lopen** .
specific school actually will school go
'This is the school year before your child actually goes to a specific school.'

(23) Vanaf dit ogenblik zijn er drie verkozenen
From this moment are there three elected
die zich fulltime met de realisatie van het
who them full time with the realisation of the
groene gedachtegoed **kunnen bezig houden** .
green idea can busy keep
'From now on there are three members elected who can work on the realisation of the green thought.'

[12]The information on word position is included in the XML structure, but is not presented in the graphical representation.

(24) Dat kristal is zo gemaakt dat het bijvoorbeeld de
that cristal is so made that it for example the
kleur rood altijd weerkaatst en niet **kan**
colour red always reflects and not can
door laten .
through let
'That cristal is made in a way that it for example always reflects the colour red and will not let it through.'

The XPath query derived from the non-interruption example *dat ze kennis moet maken* is given in (25). Note that the only difference between the two queries (20) and (25) is the first comparison operator (>).

(25)
```
//node[@cat="ssub" and node[@rel="hd"
and @pos="verb" and @begin
> ../node[@rel="vc" and
@cat="inf"]/node[@rel="svp"
and @pos="part"]/@begin] and
node[@rel="vc" and @cat="inf" and
node[@rel="svp" and @pos="part"
and @begin < ../node[@rel="hd" and
@pos="verb"]/@begin]]]
```

The query in (25) finds 94 hits in 91 sentences, of which five are presented in (26) - (30).

(26) Het zijn afspraken waar Nederland aan
it are agreements where Netherlands on
vasthoudt en waar Sint Maarten niet voor
holds and where Saint Martin not for
weg kan lopen .
away can run
'Those are deals the Netherlands hold to and which Saint Martin cannot evade.'

(27) De associates zijn jonge high potentials ,
the associates are young high potentials ,
supergedreven , die **aan komen zetten** met
super-motivated , who on come sit with
een idee .
an idea
'The associates are young high potentials, super motivated , who come up with an idea.'

(28) Daarom geeft het ministerie van VROM u tips
therefore gives the ministry of VROM you tips
en advies hoe u in huis veilig **om kunt**
and advice how you in house safely can
gaan met gas en elektra .
go with gas and electricity
'Therefore the ministry of VROM gives you tips and advice to safely handle gas and electricity.'

(29) Er is zelfs geen eensgezindheid over de
there is even no consensus on the
limieten vanaf wanneer een gas een
limits from when a gas a
broeikaseffect **teweeg zal brengen** .
greenhouse effect will bring
'There isn't even a consensus on the limits when a gas will cause a greenhouse effect.'

(30) De drie hebben dan ook te kennen gegeven dat
the three have then also to know given that
ze **zaken willen doen** , wat andere Europese
they business want to , what other European
landen weer met achterdocht vervult.
countries again with suspicion fulfills
'The three have indicated that they want to do business , which makes other European countries suspicious again.'

The results found by (17), (20) and (25) show that for this case GrETEL also finds constructions similar to the input structure. Equivalent constructions that are differently tagged are however hard to find. For example, one could argue that there are more separable verb particles present in the corpus, but GrETEL will miss them out as they are not tagged as such. In Dutch, the separable verb particle and the verb are often written as one word, cf. example (31).

(31) dat ze **moet kennismaken**
that she has acquiantance-make
'that she has to meet'

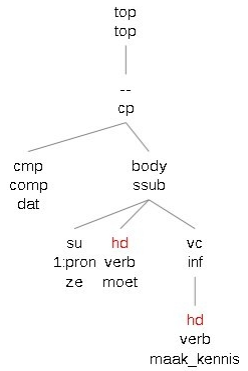

Figure 6: Alpino parse of example (31) (separable verb particle included in verb node)

The parse of sentence (31) in Figure 6 shows that the separable verb particle is included in the lemma of the verb, but in contrast to the example in Figure 5 it does not receive a tag of its own. Those sentences could be found with another query,[13] so for a more exhaustive treebank search the user is advised to use multiple input sentences.

[13] In basic search mode one should select the *lemma* button for the verb *kennismaken*. To abstract over all separable verbs, one should change the query part that refers to the separable verb `//node[@rel="hd" and @pos="verb" and contains(@root, "maak_kennis")]` to `//node[@rel="hd" and @pos="verb" and contains(@root, "_")]` in the advanced search mode, as al separable verbs have an `"_"` in their lemma.

5. Conclusion and Future Research

We have presented GrETEL, a tool for querying a treebank with natural language examples instead of a formal input query. The tool allows to search in treebanks without knowledge about the tree representations, treebank query languages, nor the specific linguistic theories in the treebank. The user provides the query engine with an example sentence, marking which parts of the sentence are the focus of the query. Through automated parsing of the example sentence and subtree extraction of the sentence part under focus the treebank is queried for the extracted subtree. GrETEL then returns sentences similar to the input example. In order to demonstrate GrETEL as a tool for treebank mining, two case studies were conducted.

Future research involves the creation of a web version of the example-based query mechanism for the LASSY treebank, integrated with the actual search mechanism, instead of the XPath query engine we currently use, backing off to more abstract subtrees, when no results are found. Furthermore, we want to include more treebanks, such as CGN and LASSY Large. Ultimately, we want to build a faster query engine, based on the Varro tree indexing toolkit (Martens, 2011), as XPath is rather slow, especially on large treebanks.

6. Acknowledgements

This paper as well as the development of the GrETEL tool is part of Nederbooms, a CLARIN project that is funded by the Flemish Community.

7. References

Gosse Bouma and Geert Kloosterman. 2002. Querying Dependency Treebanks in XML. In *Proceedings of LREC'02*, pages 1686–1691, Las Palmas, Spain.

Gosse Bouma and Geert Kloosterman. 2007. Mining Syntactically Annotated Corpora with XQuery. In *Proceedings of the Linguistic Annotation Workshop*, pages 17–24, Prague, Czech Republic.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER Treebank. In *Proceedings of TLT-1*.

Eugene Charniak. 1997. Statistical Parsing with a Context-free Grammar and Word Statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, Menlo Park. AAAI Press/MIT Press.

Walter Haeseryn, Kirsten Romijn, Guido Geerts, Jaap de Rooij, and Maarten van den Toorn. 1997. *Algemene Nederlandse Spraakkunst*. Martinus Nijhoff/Wolters Plantyn, Groningen/Deurne, second edition.

Sebastian Hellmann, Jörg Unbehauen, Christian Chiarcos, and Axel-Cyrille Ngonga Ngomo. 2010. The TIGER Corpus Navigator. In *Proceedings of TLT-9*, pages 91–102, Tartu, Estonia.

Gideon Kotzé, Vincent Vandeghinste, Scott Martens, and Jörg Tiedemann. 2012. Large Aligned Treebanks for Syntax-Based Machine Translation. In *Proceedings of LREC'12*, Istanbul.

Catherine Lai and Steven Bird. 2004. Querying and updating treebanks: A critical survey and requirements analysis. In *Proceedings of the Australasian Language Technology Workshop*, pages 139–146.

Catherine Lai and Steven Bird. 2010. LPath+: A First-Order Complete Language for Linguistic Tree Query. In *Proceedings of PACLIC 19*.

Wolfgang Lezius. 2002. TIGERSearch - ein Suchwerkzeug für Baumbanken. In Stephan Busemann, editor, *Proceedings of KONVENS-02*, Saarbrücken.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Scott Martens. 2011. *Quantifying Linguistic Regularity*. Ph.D. thesis, KU Leuven, Leuven, Belgium.

Lars Nygaard and Janne Bondi Johannesen. 2004. SearchTree - a user-friendly treebank search interface. In *Proceedings of TLT-3*, page 183–189, Tübingen.

Philip Resnik and Aaron Elkiss. 2005. The Linguist's Search Engine: An Overview. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 33–36, Ann Arbor.

Douglas L.T. Rohde, 2005. *TGrep2 User Manual*.

Leonoor van der Beek, Gosse Bouma, Robert Malouf, and Gertjan van Noord. 2002. The Alpino Dependency Treebank. In *Computational Linguistics in the Netherlands 2001*.

Ton van der Wouden, Ineke Schuurman, Machteld Schouppe, and Heleen Hoekstra. 2003. Harvesting Dutch trees: Syntactic properties of Spoken Dutch. In *Computational Linguistics in the Netherlands 2002*, pages 129–141, Amsterdam.

Frank Van Eynde. 2009. A Treebank-Driven Investigation of Predicative Complements in Dutch. In *Computational Linguistics in the Netherlands 2009*, pages 131–145, Utrecht.

Maarten van Gompel. 2011. FoLiA: Format for Linguistic Annotation. ILK Research Group, Tilburg University, September.

Gertjan van Noord, Ineke Schuurman, and Vincent Vandeghinste. 2006. Syntactic annotation of large corpora in STEVIN. In *Proceedings of LREC'06*, pages 1811–1814, Genoa, Italy.

Gertjan van Noord, Gosse Bouma, Frank Van Eynde, Daniel de Kok, Jelmer van der Linde, Ineke Schuurman, Erik Tjong Kim Sang, and Vincent Vandeghinste. in press. Large Scale Syntactic Annotation of Written Dutch: Lassy. In *Essential Speech and Language Technology for Dutch: resources, tools and applications*. Springer.

Gertjan van Noord. 2006. At Last Parsing Is Now Operational. In *TALN 2006*, pages 20–42.

Holger Voormann and Wolfgang Lezius. 2002. TIGERin - Grafische Eingabe von Benutzeranfragen für ein Baumbank-Anfragewerkzeug. In Stephan Busemann, editor, *Proceedings of KONVENS-02)*, Saarbrücken.

Jan Štepánek and Petr Pajas. 2010. Querying Diverse Treebanks in a Uniform Way. In *Proceedings of LREC'10*, Valletta, Malta.