LASSY WORK PACKAGE 5

FEASIBILITY STUDY SEARCH / EXTRACTION TOOLS

Deliverable 5.1

Erik Tjong Kim Sang
Alfa-informatica
University of Groningen

LASSY WP 5.1: Feasibility Study Search / Extraction Tools

Erik Tjong Kim Sang
Alfa-informatica, University of Groningen

24th March 2009

1 Introduction

We examine existing tools for exploring corpus annotation layers. We look at dedicated tools as well as general languages for handling data encoded in XML.

2 Corpus annotation

We assume that the annotation layers in our text corpora are made up of XML structures. XML (eXtensible Markup Language) is a standard language for encoding the structure of text. Here is an example of data encoded in XML:

```
<sentence>Zo ziet u !</sentence>
```

The data consists of a single sentence containing three words and a punctuation sign. The sentence is preceded by the tag `<sentence>`, which indicates the start of a sentence, and is followed by the tag `</sentence>` which indicates a sentence end. Tags can also contain attributes:

```
<node id="4" pos="noun">u</node>
```

The word *u* is an XML element of type node. The start node tag (`<node...>`) contains two attributes: an id value (4) and a part-of-speech value (noun). Attributes do not need to be repeated in the closing tag (`</node>`).

```
<alpino_ds version="1.2">
 <node cat="top">
   <node cat="smain">
     <node cat="np">
       <node pos="det" word="De"/>
       <node pos="noun" word="wielophanging"/>
     </node>
     <node pos="verb" word="heeft"/>
     <node pos="noun" word="schroefveren"/>
     <node pos="adv" word="rondom"/>
   </node>
   <node pos="punct" word="."/>
 </node>
 <sentence>De wielophanging heeft schroefveren rondom .</sentence>
</alpino_ds>
```

Figure 1: Simplified syntax tree for the Dutch sentence *De wielophanging heeft schroefveren rondom.* Apart from the words, the tree includes word classes (`pos`) and phrases and their categories (`cat`). Tags like `<abc/>` do not require an additional closing tag.

The corpus data which we will use in this report, will be encoded in XML. The structures will be more complex than in the preceding examples. Tags will include more attributes, typically between 5 and 10. Entities can be embedded in other entities. A simplified example of a syntax tree for a Dutch sentence can be found in Figure 1.

3 Corpus queries

What kind of queries could be submitted to a text corpus with syntactic annotations? We present three corpus studies as well as their research questions.

3.1 Tjong Kim Sang (2009)

[Tjong Kim Sang, 2009] searches in an annotated text corpus for phrases that indicate a hypernym relation. A hypernym of word X is a word which describes words like X and has a broader meaning. For example, *color* is a hypernym of *red*. An example of a phrase indicating a hypernym relation is *furniture such as table*. From this phrase, we can derive that *furniture* is a good candidate hypernym for *table*.

Two basic queries are used for extracting the phrases from the corpus:

- find two given nouns linked by a short path (three nodes or fewer) in a syntax tree
  for example: *furniture/N* words+ *table/N*
- find a given sequence of syntax tree nodes with a noun to the left and right
  for example: N_1 *such that* N_2

The queries are formulated in XQuery as a search algorithm. In order to save processing time, the two queries are combined into one: find a pair of arbitrary nouns linked by a short path in a syntax tree. The search query specified in XQuery looks like a procedural algorithm: every pair of nodes in a sentence tree is inspected. If both nodes contain a noun and if they are linked by a short path then the pair and the path are selected. Further processing of the selected combinations of noun pairs and paths does not require access to the syntax trees.

3.2 Van Noord (2009)

[Van Noord, 2009b] presents a method for improving the Alpino parser for Dutch by adding additional lexical relations to the part of the parser that selects the best syntactic tree for a sentence. The lexical relations are extracted from a large parsed corpus which contains many dependency relations. The main extraction query used in this study is quite simple:

- find any dependency relation between two words

This basic query already generates many interesting lexical relations, for example that *milk* is generally an object of *to drink* and not a subject. However, in some cases a singular dependency relation is unable to express the relation between words. Examples of this for verb-object relations are objects containing a conjunction (*we caught spiders and flies*) and verbs inside a relative clause (*the diamonds which Dennis stole*). Two additional queries have been defined for collecting the relevant relations:

- find all noun phrases with a head that is a conjunction with a verb as head and return the noun phrase heads and the verb
- find all verbs that have a relative pronoun in the object position and return the head of the relative pronoun and the verb

The lexical relations extracted with the queries proved to be useful for the parser. With the relations added to the disambiguation model, the parser made 3% fewer errors.

3.3 Bouma and Spenader 2009

[Bouma and Spenader, 2009] study the distribution of strong and weak reflexive pronouns in Dutch. Reflexive pronouns are words like *myself* and *himself*. Dutch has two forms of the third person reflexive pronoun: *zichzelf* and *zich*, both of which could be translated as either *himself*, *herself* and *themselves* in English. The two forms are distributed differently for different verb groups.

The goal of this study is to find support for existing theories about the distribution of reflexive pronouns. For this purpose, occurrences of reflexive pronouns and associated verbs are collected from a large parsed corpus. The following query is used for obtaining interesting material:

- find all verbs with a third person object

The results are used for computing the frequency of the occurrences of each verb with a reflexive pronoun as object. In order to adjust for verb ambiguity, frequencies are computed for verbs in combination with their subcategorization frame. Additionally 20% of the data which contained ambiguous cases, were discarded. The study showed that in about half of the cases, the distribution of weak and strong reflexive pronouns for a certain verb could be predicted from the frequency of which any reflexive occurred with the verb.

4 Tools for linguistic analysis

In this section we examine different available tools for manipulating natural language annotations which are encoded in XML. We are especially interested in tools that can handle the queries which were used in the case studies described in the previous section.

4.1 TigerSearch

TigerSearch is a software suite developed for inspecting and querying text corpora with linguistic annotations [König et al., 2003]. It accepts different input formats which are converted to an internal format. TigerSearch includes a graphical interface which enables browsing the syntactic trees in the corpus and selecting them with queries. These queries can be expressed graphically as well as in a query language which involves boolean expressions containing name-value pairs for syntactic features.

We were unable to express in TigerSearch all queries required for the case studies but here are some approximations:

1. #n:[cat="NP"]&#o:[cat="NP"]&#n.1,3#o

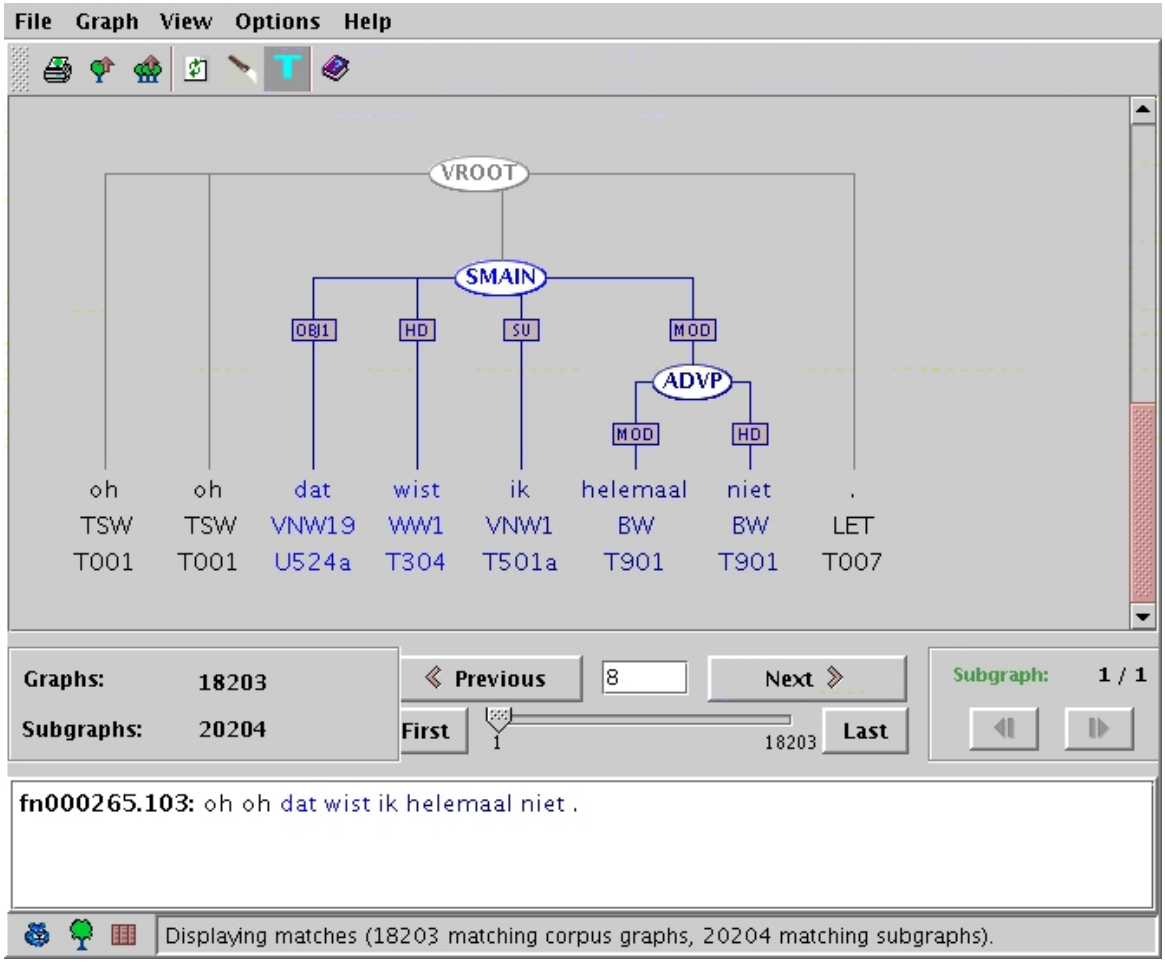

Figure 2: Screenshot of the TigerSearch browser. It shows a result for query 3 (verbs with an object).

2. [cat="SV1"]&OBJ1[cat="CONJ"]
3. #s:[cat="SMAIN"]>OBJ1#n&#s>HD#v

The queries contain several distinct parts. [cat="XP"] searches for a node with category XP. These nodes can be assigned a name starting with # by placing the name in front of them separated by a colon. A query can involve searching for more than one node by combining them with ampersands (AND). The results will only contain parse trees which contain all required nodes. There are two additional operators: . stands for concatenation and > for domination

The first query finds noun phrases which are separated by a short phrase containing at most three words (from the study described in [Tjong Kim Sang, 2009]). However, this is not what we were interested in. We wanted to find noun phrases separated by at most three *nodes*. We were unable

to express this with TigerSearch.

[Van Noord, 2009a] was among others interested in conjunctions in object positions. The second query is an example of how these could be retrieved from the corpus. It is incomplete with respect to the required category values of the parent of the verb (there are more options). It is too relaxed with respect to the contents of the conjunction phrases. Conjunctions involving nonnoun phrases should be excluded.

Verbs with objects can be retrieved with the third query. [Bouma and Spenader, 2009] were interested in third-person objects. In the corpus with which TigerSearch is provided (Corpus Spoken Dutch), morphological information is hidden in a morphological tag. The query can easily be adapted to retrieve only words with the proper morphological tag. However, many different tags are acceptable and expanding the query would involve specifying each and every of them.

4.2 dtsearch and XPath

dtsearch is a program for searching in a collection of XML documents developed at the University of Groningen[1]. It uses XPath for specifying queries. XPath is an XML query language developed by the World Wide Web Consortium (W3C) and released in 1999. The language contains instructions for selecting nodes in an XML tree by name, position or attributes. For example: `/X/Y` selects all nodes `Y` which are a child of the top node `X`, and `//Z[@a="b"]` selects nodes `Z` with an attribute `a` that has value `b`.

XPath is a very useful query language. Here are XPath expressions that approximate the three case study queries:

1. //*[count(*[@cat="np"])>1]/*[@cat="np"]
2. //*[@cat="conj" and ./*[@cat="np"] and ../*[@pos="verb" and @rel="hd"]]
3. //*[./*[@pos="verb"] and ./*[@rel="obj1"]]/*[@pos="verb" or @rel="obj1"]

The first query attempts to retrieve distinct noun phrases which are linked by a path of three nodes or less. The query focuses on a particular subset of such phrase pairs: the noun phrases that share a parent. These are linked by a single-node path. Seperate queries are required for finding node pairs linked by longer paths. However, nodes found by such queries need to be restricted to non-ancestor pairs. We do not know how to add this additional restriction to the XPath query. This requires nodes in the query to be named but as far as we can see this is not possible in XPath.

The goal of the second query is to find conjunctions of noun phrases in an object position of a verb. There are two problems with this query. First, it selects and outputs the conjunction node

[1] An unrelated text retrieval system dtsearch® also exists.

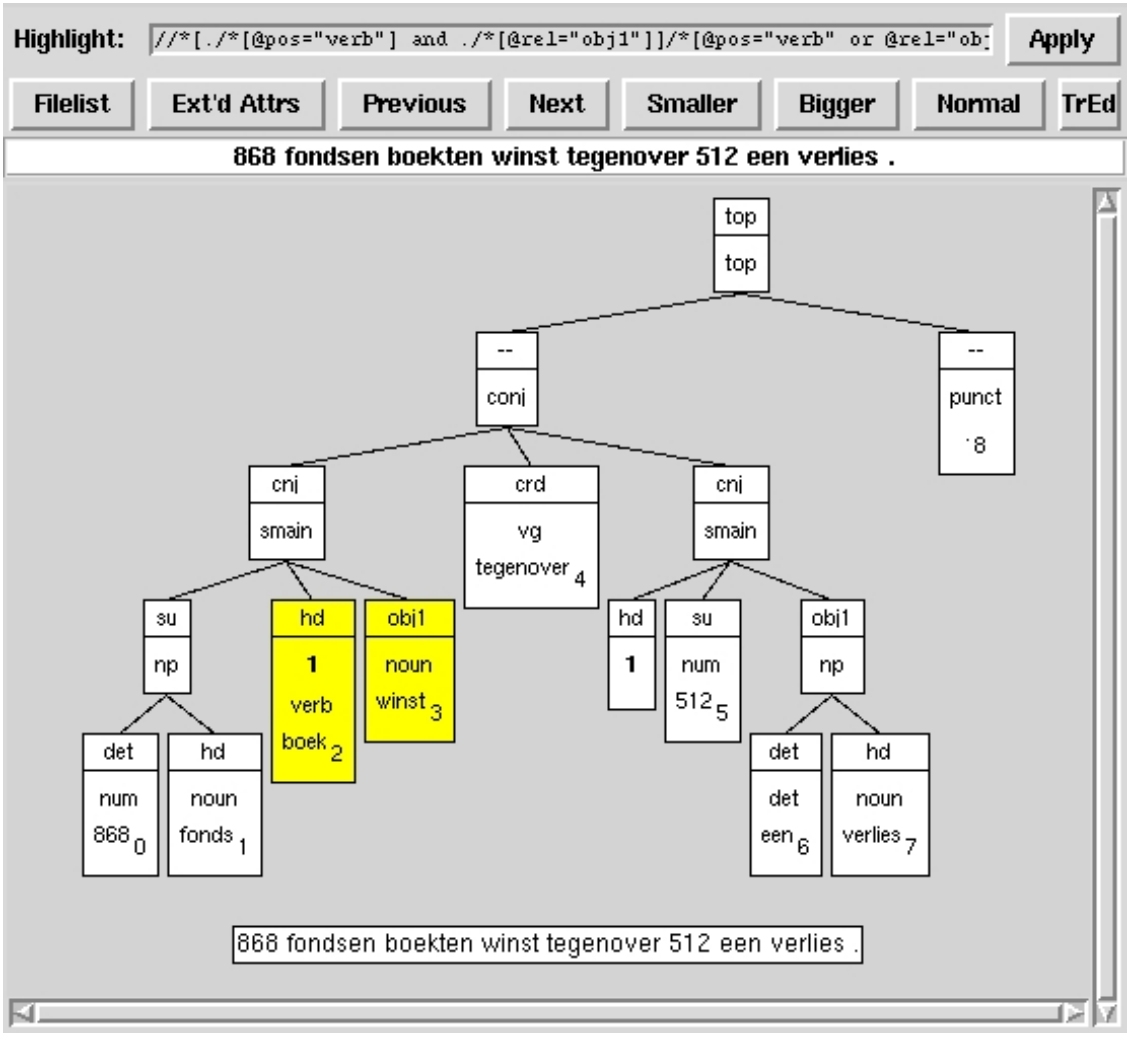

Figure 3: Screenshot of the dtsearch browser. It shows a result for query 3 (verbs with an object).

rather than the embedded noun phrases and the sister verb node. We do not know how to select the required nodes. The second problem is that the query might fail to retrieve all relevant data from our target corpus. In this corpus, the conjunction might be stored in a separate part of the tree with an index link to the object position. In order to find the conjunction, we need to be able to match the two index values, a problem which have been unable to solve in XPath.

The third query finds pairs of verbs and their associated objects. As far as we can see, this query works fine. It looks for nodes with both a verb child and an object child and return all children of the node that satisfy one of these criteria.

```
(VP (VBD contracted) (NP (JJ asbestos-related) (NNS diseases)))
(VP (VBG waiving) (NP (NN management) (NNS fees)))
(VP (VB obtain) (NP (JJ regulatory) (NN approval)))
(VP (VBZ employs) (NP (CD 2,700) (NNS people)))
```

Figure 4: Examples of the output of TGrep2 for query 3 (verbs with an object).

4.3 TGrep2

TGrep2 is a tool for analyzing bracketed corpora such as the Penn Treebank [Marcus et al., 1993]. The tool was developed by Douglas Rohde in 2001 [Rohde, 2005] and was based on the TGrep tool by Richard Pito. Before processing, the bracketed corpora need to be compiled to a binary format:

```
sed 's/^(/(TOP /' wsj*mrg > corpus.wsj; ./tgrep2 -p corpus.wsj corpus.t2c
```

TGrep2 has an extensive query language which allows specification of ancestor relations, order and lexical content. It is also possible to label temporary results. Here are the three example queries written down in the TGrep2 query language:

1. `NP > (* < (NP $.. NP))` (with TGrep2 options -a and -s)
2. `VP < (NP < CC)`
3. `VP < NP`

The first query attempts to retrieve noun phrases that are linked by a short dependency path. It focuses on noun phrases that share a parent. It searches for two consecutive noun phrases (`$..`), selects their parent (`* <`) and returns all the noun phrase children within the parent (`NP >`). TGrep requires two extra command line options here: -a for returning all matching phrases rather than only the first and -s for marking the first match of each sentence. However, if one sentence contains more than one matching group then this single label will not be enough to separate the groups from each other.

The second query finds all verb phrases (`VP`) which dominate a noun phrase containing a conjunction (`NP < CC`). This query contains two problems. First, the noun phrase will not always be a direct object. However, the target corpus (Penn Treebank) used for this text query does not contain object information and this is the closest we came to finding objects. Second, the query returns the complete verb phrase. It is possible to extract the relevant verbs and nouns from the phrase but this requires two separate queries. The target corpus does not consistently mark all noun phrases and this complicates collecting the relevant nouns and noun phrases.

```
38 VP: 38 VP, 41 NP
(38 VP (39 VBD contracted) (41 NP (42 JJ asbestos-related) (44 NNS diseases)))
33 VP: 33 VP, 36 NP
(33 VP (34 VBG waiving) (36 NP (37 NN management) (39 NNS fees)))
33 VP: 33 VP, 36 NP
(33 VP (34 VB obtain) (36 NP (37 JJ regulatory) (39 NN approval)))
7 VP: 7 VP, 10 NP
(7 VP (8 VBZ employs) (10 NP (11 CD 2,700) (13 NNS people)))
```

Figure 5: Examples of the output of Corpussearch2 for query 3 (verbs with an object). They contain index numbers plus a complete phrase. The relevant parts have to be extracted by other means.

The third query looks for verbs and their objects. It also suffers from the first problem mentioned in the previous paragraph: it matches direct objects and some other phrases. For example, the query will return two pairs for the sentence *John gave Bill the book*: *gave* together with *Bill* and *gave* in combination with *the book*.

4.4 Corpussearch2

Corpussearch2 is a Java program for searching in text corpora annotated according to the format of the Penn Treebank [Randall, 2005]. It includes a query language which is based on the relations of nodes in syntactic trees. The relations have been based on linguistic relations. The most important relations in the Corpussearch2 query language are Dominates and CCommands.

Here are approximations of the three case study queries in Corpussearch2:

1. (NP HasSister NP)
2. (VP iDominates [1]NP) and ([1]NP iDominates CC)
3. (VP iDominates NP)

A general problem with Corpussearch2 is that target phrases cannot easily be extracted from the output. The results contain large phrases with a separate list of numbers indicating the target phrases (see Figure 5). In order to obtain the target phrases, the number list and the large phrases have to be combined in some way.

The first query finds noun phrases that are linked by a small number of nodes. It focuses on sister nodes (which are linked by a single parent). Here the output format enables retrieval of all relevant pairs (with some extra work). However, the encoding format of the source corpus (Penn Treebank) does not make possible retrieval of all relevant material because noun phrases of one word have

```
<pair object="zoon" verb="heb"/>
<pair object="studie" verb="zet_voort"/>
<pair object="leerstoel" verb="ontvang"/>
<pair object="diploma" verb="behaal"/>
<pair object="toespraak" verb="woon_bij"/>
<pair object="indruk" verb="maak"/>
<pair object="Berlijn" verb="verlaat"/>
<pair object="opdracht" verb="krijg"/>
<pair object="mei_viering" verb="geef_vorm"/>
<pair object="zoeklicht" verb="laat"/>
```

Figure 6: Examples of the output of XQuery for query 3 (verbs with an object).

not been labeled as noun phrase. Like with the previous tools, for each relevant path of relevant length, at least one other query needs to be created in Corpussearch2.

Queries 2 and 3 demonstrate how objects can be retrieved. The restriction mentioned in the TGrep2 section also applies here: in the source corpus, verb objects have not been marked. The two queries will retrieve the relevant verb objects but they will also retrieve some noise. In the queries, iDominates is used for requiring immediate domination (parent-child). Query 2 shows that phrases can be labeled, a useful option.

4.5 XQuery

XQuery is an XML query language developed by the World Wide Web Consortium (W3C) and released in 2007 [Boag et al., 2007, Brundage, 2004]. It offers the same query facilities as XPath with an additional functional programming language. This makes XQuery very useful for defining complex XML queries. Here is an example for the first case study query:

```
let $nodes := $top//node[@cat="np"]
for $np1 in $nodes
   for $np2 in ($np1/../node, $np1/../../node, $np1/../node/node)
      let $head1 := get-head($np1)
      let $head2 := get-head($np2)
      return
         if ($np2/@cat eq "np")
         then <pair head1="{$head1/@root}" head2="{$head2/@root}" />
         else ""
```

This program starts with collecting all nodes with category label "np". For each of these nodes, it collects all non-ancestor nodes that are linked to the node via a path of one or two nodes. Nodes

that also have a category label "np" are used for generating output pairs. The output contains the head words of the first and the second noun phrase. The heads are generated by a user-implemented function `get-head()`. An additional function is necessary for collecting information about the path linking the two nodes (left out of this example).

The program for retrieving verb-object pairs of the third case study is similar:

```
let $nodes := $top//node[(@pos="verb")]
for $verb in $nodes
   for $object in $verb/../node
      let $head0 := get-head($object)
      return
         if (exists($object/@rel) and $object/@rel eq "obj1")
         then <pair verb="{$verb/@root}" object="{$head0/@root}" />
         else ""
```

It collects all nodes with the part-of-speech tag "verb". Then it outputs pairs containing these nodes and sister nodes that have a relation label "obj1". The program for collecting information about verb-object pairs where the object contains a conjunction, is an extension of this:

```
let $nodes := $top//node[(@pos="verb")]
for $verb in $nodes
    for $object in $verb/../node
       for $child in $object/node
          let $headC := get-head($child)
          return
             if (exists($object/@rel) and $object/@rel eq "obj1" and
                 exists($object/@cat) and $object/@cat eq "conj" and
                 exists($child/@rel) and $child/@rel eq "cnj")
             then <pair verb="{$verb/@root}" object="{$headC/@root}" />
             else ""
```

This time the results contain the verb and the head words of the children of the object node, provided that these children are part of a conjunction.

The example queries show that XQuery enables encoding complex queries.

5 Concluding remarks

We have evaluated five different tools for exploring syntactically annotated corpora. Each of the tools has been used to implement three different queries for syntactic information. Two of the tools (TGrep2 and Corpussearch2) do not process XML data but use the Penn Treebank format, which is less useful for our purposes. XPath does not allow labeling and re-use of intermediate results. TigerSearch and XQuery seem to be most useful. The flexibility of XQuery makes this tool/programming language the most interesting option for our research project LASSY.

However, the complexity of XQuery also has a drawback: it is not suitable for novice users. If we want to offer access to the LASSY corpora through XQuery, we should also provide access via a user-friendly tool for users that have not learned how to program computers. Such a tool would translate frequent general queries from some kind of user language to a set of XQuery commands, execute these and enable saving the results.

Care should be taken that frequently issued linguistic queries can easily be coded in XQuery. In our opinion this does not involve extending XQuery, as we do not have the expertise for this, but careful definition of the XML structures used in our annotated corpora. One of the steps already set in this direction is the inclusion of begin and end attributes to allow for queries with positional information.

The success of a future LASSY corpus exploration tool will also depend on the possibility of having naive users testing the software. Our ongoing cooperation with other researchers will hopefully provide the necessary test group.

References

[Boag et al., 2007] Boag, S., Chamberlin, D., Fernández, M. F., Florescu, D., Robie, J., and Siméon, J. (2007). *XQuery 1.0: An XML Query Language*. World Wide Web Consortium (W3C). http://www.w3.org/TR/xquery/.

[Bouma and Spenader, 2009] Bouma, G. and Spenader, J. (2009). The distribution of weak and strong object reflexives in dutch. In *Proceedings of TLT7*. LOT, Groningen, The Netherlands.

[Brundage, 2004] Brundage, M. (2004). *XQuery – The XML Query Language*. Addison-Wesley.

[König et al., 2003] König, E., Lezius, W., and Voormann, H. (2003). *TIGERSearch 2.1 - User's Manual*. IMS, University of Stuttgart.

[Marcus et al., 1993] Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):331–330.

[Randall, 2005] Randall, B. (2005). *CorpusSearch 2: a tool for linguistic research*. University of Pennsylvania. http://corpussearch.sourceforge.net/.

[Rohde, 2005] Rohde, D. L. (2005). *TGrep2 User Manual*. MIT Technical Report. version 1.15.

[Tjong Kim Sang, 2009] Tjong Kim Sang, E. (2009). To use a treebank or not – which is better for hypernym extraction. In *Proceedings of the Seventh International Workshop on Treebanks and Linguistic Theories (TLT 7)*. Groningen, The Netherlands.

[Van Noord, 2009a] Van Noord, G. (2009a). Huge parsed corpora in lassy. In *Proceedings of TLT7*. LOT, Groningen, The Netherlands.

[Van Noord, 2009b] Van Noord, G. (2009b). Self-trained bilexical preferences to improve disambiguation accuracy. In *Selected Papers from the IWPT 2007, CONLL 2007 and IWPT 2005*. Springer.