Refining Queries on a Treebank with XSLT Filters. Approaching the Universal Quantifier [*]

George Smith

Universität Potsdam

This paper discusses the use of XSLT stylesheets as a filtering mechanism for refining the results of user queries on treebanks. The discussion is within the context of the TIGER treebank, the associated search engine and query language, but the general ideas can apply to any search engine for XML-encoded treebanks. It will be shown that important classes of linguistic phenomena can be accessed by applying relatively simple XSLT templates to the output of a query, effectively simulating the universal quantifier for a subset of the query language.

1 Introduction

In the TIGER treebank (Brants et al., 2002), syntactic structure is encoded via restricted directed acyclic graphs, henceforth syntax graphs. These are tree-like structures with potentially crossing branches and labelled edges. The corpus is available in XML format. The search engine TIGERSearch (König et al., 2003) was developed to enable linguists with no previous experience in the use of either query languages or XML-encoded corpora to work with the corpus. In TIGERSearch, the encoded structures are presented to the user in a graphic representation familiar to this particular constituency. A specially designed language (König and Lezius, 2003) allows the user to query the treebank using concepts already familiar to linguists: immediate dominance, linear precedence and derived relations. The formulation of a query involves describing desired structural characteristics. The result of a query is the set of all structures in the corpus which have those characteristics.

[*] I would like to thank Esther König and Wolfgang Lezius for numerous helpful discussions relating to the TIGER query language. This paper is dedicated to Peter Eisenberg on the occasion of his 65th birthday.

Interdisciplinary Studies on Information Structure 02 (2005): 117–128

Dipper, S., M. Götze and M. Stede (eds.):

Heterogeneity in Focus: Creating and Using Linguistic Databases

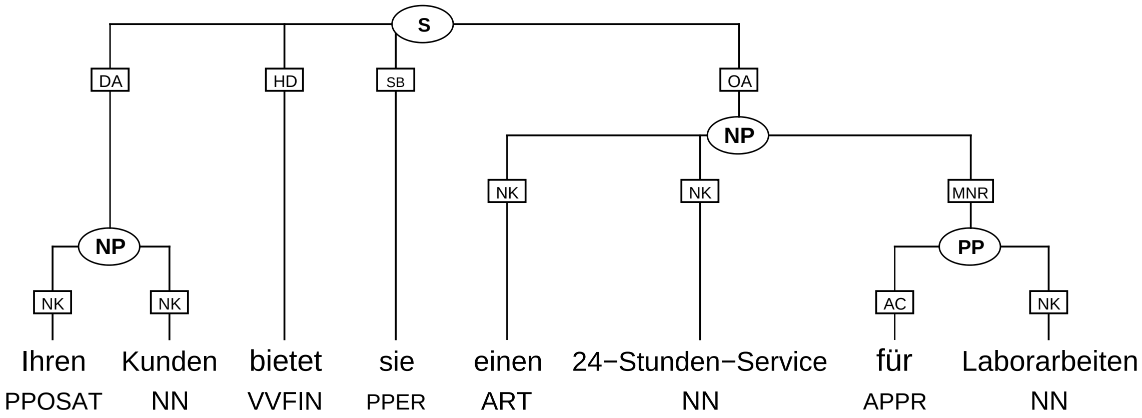

Figure 1: Three Place Verb

2 Structural Search

This section will demonstrate the importance of a universal quantifier in a treebank query language. A brief overview of one of the main mechanisms for searching for syntactic structure using the TIGER language will be provided in 2.1. One important area of syntax which could be more easily investigated via a universal quantifier will be given in 2.2 and 2.3.

2.1 Immediate Dominance

This section will provide a brief overview on searching syntactic structures involving immediate dominance.

(1)	Ihren	Kunden	bietet	sie	einen	24-Stunden-Service	für Laborarbeiten
	her	customers	offers	she	a	24-hour service	for lab work
	‘She offers her customers a 24-hour-service for lab work.’						

The sentence in example (1) has the graphical representation in figure 1. A user could be interested in various structural characteristics of the sentence.

Several simple queries which match structures in the sentence are given in (2).

(2) a. `[cat="S"] >SB [pos="PPER"]`
b. `[cat="NP"] >MNR [cat="PP"]`
c. `[cat=("S"|"VP")] >DA []`

The query (2-a) matches all sentences in which the subject is a personal pronoun. This is accomplished by describing two nodes and a relation between them. The expression `[cat="S"]` describes a node with the value `S` (sentence) for the feature `cat` (category). The expression `[pos="PPER"]` describes a node with the value `"PPER"` (pronoun, personal) for the feature `pos` (part of speech). The operator `>` defines a relation of immediate dominance in which the node described to the left dominates the node described to the right. This relation is labelled `SB` (subject). The label indicates the function which the child node has within the constituent formed by the parent node.

Similarly, the query (2-b) matches all structures in which a noun phrase has a prepositional attribute, that is, in which a node with the value `"NP"` (noun phrase) for the feature `cat` immediately dominates a node with the value `"PP"` (prepositional phrase) for the feature `cat` and the relation of immediate dominance between them is labelled `MNR` ("modifier nominal right": modifier of a noun, to the right).

The final query (2-c) matches any structure in which a node with either the value `"S"` or `"VP"` (verb phrase) for the feature `cat` dominates a node which is not further specified, and the relation of immediate dominance between them is labelled `DA` (dative), indicating that the child node functions as a dative argument within the constituent formed by the parent.

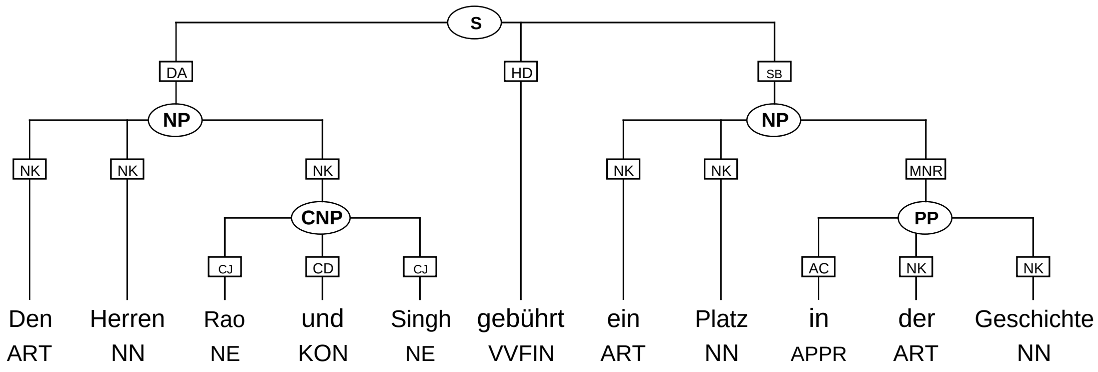

Figure 2: Two Place Dative Verb

2.2 Argument Structure and Agentivity

This section will discuss the argument structure of two classes of German verbs with regard to agentivity, preparing the argument made in section 2.3 that a universal quantifier is an important part of a treebank query language.

(3) Den Herren Rao und Singh gebührt ein Platz in der Geschichte
the Messrs Rao and Singh deserve a place in the history
'Messrs Rao and Singh deserve a place in history.'

Let us now take a look at another sentence, given in (3) with the graphical representation in figure 2, which bears certain similarities to the sentence in (1). Both sentences contain an argument in dative, which can be accessed by query (2-c). They also exhibit important differences with regard to argument structure, which will be explored here with reference to ideas presented in Primus (1999) and Eisenberg (2004), in which arguments are seen as having varying degrees of agentivity and patientivity. Agentive and patientive properties of an argument are determined compositionally using a set of thematic roles presented here in order of declining agentivity: *control*, *cause*, *move*, *exper*, *possess*.

Example (1) has the prototypical structure of a sentence with a three-place verb from the semantic field of giving and taking. The argument with the highest degree of agentivity is encoded in nominative. It is a prototypical agentive subject, excercising control, causing movement. The argument encoded in accusative is a prototypical patient. The argument encoded in dative has a low degree of agentivity, here exibiting a potential for possession. This weak agent is a prototypical recipient.

Example (3) on the other hand has a rather different argument structure. There is no argument with strong agentive properties, excercising control, or being a cause. The argument with the highest degree of agentivity is a possessor. This weak agent or recipient is again encoded in dative. The argument with no agentive properties, the patient, is encoded in nominative, avoiding a sentence with no subject. Interesting classes of German verbs have this type of argument structure, with a patientive subject and an argument in dative exhibiting a degree of agentivity compatible with the recipient role, one important group being the psychological verbs, which have the slightly more agentive *exper* encoded in dative.

2.3 The Need for the Universal Quantifier

While the query in (2-c) is sufficient for a user who simply wishes to find arguments which are dative recipients as it will access both (1) and (3), a user who wishes to capture the differences between the argument structures of the two sentences will need to further refine the query. To search for sentences with three-place verbs such as in (1) is simple.

(4)
```
#p:[cat=("S"|"VP")] >DA [] &
#p >OA []
```

The query in (4) uses a variable #p to extend the query (2-c). It specifies that there is a node #p which immediately dominates one node which has the function DA as well as another which has the function OA (object, accusative).

(5)
```
#p:[cat=("S"|"VP")] >DA [] &
#p !>OA #c
```

It is then possible to specify the presence of an argument in acccusative. In the current implementation of the TIGER language it is, however, not possible to specify the absence of one. Simply negating this relation of dominance, as in (5), results in a query stating that in addition to the dative dominated by #p, there is also a node #c, and that the specified relationship of labelled immediate dominance does not hold between #p and #c. The node #p may dominate the node #c, in which case the edge label must not be OA, or #p may not dominate #c at all. It can be any node in the tree for which the specified labelled dominance relation between #p and #c does not hold.

One option for accessing such structures might be to search using predefined classes of verbs which can have the specified argument structure. In TIGERSearch it is in principle possible to define a set of lexemes in a template (König et al., 2003) and then use that template in a search. But this method would be indirect at best, not taking the potential for multivalency into account. It also presupposes that the user already has a list of all verbs in the corpus which can have the relevant argument structure, whereas the user may well be interested in finding verbs which can have that structure. Besides, there are other important types of structures which can only be specified by stating that a constituent of a particular type has no children with particular characteristics. Examples would be sentences with no subject, including the impersonal passive, and noun phrases which are lacking a determiner, or have no attributes of a specific type, etc. There is clearly a need for a general mechanism which can accomplish this.

(6)
```
exists #p: forall #c: ((#p > #c) =>
                              (#p !>OA #c))
```

The most comfortable option for the user would be a possible extension of the TIGER language which is discussed in König et al. (2003), the addition of the universal quantifier, exemplified here in (6). This query specifies that there exists a node `#p` such that for all nodes `#c` which are immediately dominated by `#p`, the dominance relation is not labelled `OA`. The computational expense incurred by the universal quantifier is noted as the reason why it has not (yet) been included in the language. It is indeed clear that efficiently implementing a full universal quantifier as a part of the language would be difficult and time consuming. Another option, discussed in section 3, would be to filter the results of user queries such that the undesired structures are removed.

3 Simulating the Universal Quantifier with XSLT Stylesheets

This section will discuss the use of XSLT stylesheets to filter the results of user queries as a means of finding structures which would otherwise require that a universal quantifier apply to a relation of immediate dominance. XSLT was chosen for two reasons: The filtering can be accomplished with a single template and a varying test. TIGERSearch already includes an interface for piping the output of queries through XSLT. The XML representation of the syntactic structures involved will be described in 3.1. Specific filters will be described in 3.2.

3.1 The XML Representation of Syntactic Structure

In this section we will explore those elements of the XML output of a search which are relevant for further processing via an XSLT filter.

(7)
```
#p:[cat=("S"|"VP")] >DA #da
```

(8)
```
<match subgraph="s60_505">
  <variable name="#p" idref="s60_505" />
  <variable name="#da" idref="s60_503" />
</match>
```

The query in (7) is a variation of the query (2-c) from section 2.1 with variables. If we run this query on the TIGER corpus and export the results as an XML file we find, among other structures, a set of matches. These matches are encoded as in (8). A match element contains a pointer to the matching subgraph (the `subgraph` attribute). Here we see that the element `match` has children of type `variable`. These elements contain pointers to respective nodes of constituent structure (the `idref` attribute). The matching nodes are given in (9) and (10)

(9)
```
<nt id="s60_505" cat="S">
  <edge label="DA" idref="s60_503" />
  <edge label="HD" idref="s60_6" />
  <edge label="SB" idref="s60_504" />
</nt>
```

(10)
```
<nt id="s60_503" cat="NP">
  <edge label="NK" idref="s60_1" />
  <edge label="NK" idref="s60_2" />
  <edge label="NK" idref="s60_500" />
</nt>
```

The individual non-terminal nodes have an atttribute `cat` which encodes their grammatical category, as well as child elements of type `edge`, which represent the edges pointing to their respective child nodes. The `edge` nodes themselves have an idref attribute which points to the respective child nodes, as well as a label attribute, which indicates the function of the child node within the constituent formed by the parent node.

3.2 Filtering Matches with XSLT

This section will describe the use of an XSLT template which functions as a filter, blocking matches which do not meet certain requirements. First we will examine filters which remove matches in which a node #p has children with a particular syntactic function, then we will examine a filter which removes matches which have children with a particular syntactic category.

(11)
```
<xsl:variable name="test">
  0=count(key('idkey',
              variable[@name='#p']/@idref)
             /edge[@label='OA'])
</xsl:variable>
```

(12)
```
<xsl:template match="match">
  <xsl:if test="xalan:evaluate($test)">
    <xsl:apply-templates select="ancestor::s"
                         mode="print">
    <xsl:with-param name="matchroot"
                    select="@subgraph"/>
    </xsl:apply-templates>
  </xsl:if>
</xsl:template>
```

The template in (12) could be applied to the output of the query (7). It filters out matches in which the node #p has children which function as OA. Undesired results are filtered by applying a test $test before printing. Examples of XSLT stylesheets that print sentences are included with TIGERSearch (König et al., 2003). The work here is done in the filter[1]. The filter checks to make sure that the node pointed to by `#p` (namely `variable[@name='#p']/@idref`) has no children of type `edge` with the value `OA` for the attribute `label`. The test string can then be varied with regard to the name of the variable to be checked and the function(s) to be excluded. A variation on the test which would also exclude matches with object clauses (`OC`) is given in (13).

(13)
```
<xsl:variable name="test">
  0=count(key('idkey',
              variable[@name='#p']/@idref)
             /edge[@label='OA'or @label='OC'])
</xsl:variable>
```

Filters removing matches in which a node has no children with a particular syntactic category are more complex due to the need for an additional pointer.

(14)
```
<xsl:variable name="test">
  0=count(key('idkey',
              (key('idkey',
                   variable[@name='#p']/@idref)
                  /edge/@idref))
              [@pos='ART'])
</xsl:variable>
```

The filter in (14) could be applied to the matches of a query in which `#p` is bound to nodes with the syntactic category NP, to filter out noun phrases which do not have an article. The inner pointer is structured analog to that in (11) and (13). The outer pointer locates child nodes. The predicate `[@pos='ART']`

[1] The use of the Xalan extension function `evaluate` is not crucial here, but does make the code more modular and thus easier for the inexperienced user to modify.

locates those children with the value `ART` (article) for the attribute `pos`.

(15)
```
exists #p: forall #c: ((#p > #c) =>
                          (#c:[pos!="ART"]))
```

At this point it becomes excruciatingly clear that the restriction that a node have no children with a particular syntactic category or a particular syntactic function is far better stated with a representation as in (15), in the type of representation envisioned by König et al. (2003) than it is in raw XML.

4 Conclusion and Directions for Further Research

This paper has shown that relatively simple XSLT stylesheets are capable of providing important functionality needed by linguists interested in types of syntactic structure best described by stating that a node of a particular type has no children of a particular type or with a particular function. While the XML structures and XSLT code presented here is simple from a programming standpoint, the constituency for whom TIGERSearch was developed generally lacks the experience in computer science necessary to formulate or even modify example stylesheets. Indeed, the use of data abstraction, the graphical representation of syntactic structure as opposed to the raw XML representation, as well as the development of a specialized query language based on linguistic concepts as opposed to suggesting that linguists access the corpus via a generic XML solution such as XPath, XSLT or XQuery, was predicated on the idea that a treebank can only gain widespread use within the linguistic community if that community can query the treebank using tools it is comfortable with.

Further research could be directed toward building a graphical user interface which would take expressions formulated in the representation of the universal quantifier described in König et al. (2003) and create an XSLT template filter on the fly. This could be a stand alone application, or it could be integrated in

TIGERSearch. While extending the universal quantor to the immediate dominance relation would be by far the most useful type of extension, the idea could be expanded to other relations, such as the relation of linear precedence. This type of a solution would be less than the more elegant solution involving a full implementation of the universal quantifier, but it would provide a good deal of functionality and would be easier to implement.

Bibliography

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, Sozopol, 2002.

Peter Eisenberg. *Grundriß der deutschen Grammatik. Der Satz*. Metzler, Stuttgart, 2nd edition, 2004.

Esther König and Wolfgang Lezius. The TIGER language - a description language for syntax graphs, Formal definition. Technical report, 2003.

Esther König, Wolfgang Lezius, and Holger Voormann. *TIGERSearch User's Manual*. IMS, University of Stuttgart, Stuttgart, 2003. URL `http://www.tigersearch.de`.

Beatrice Primus. *Cases and Thematic Roles. Ergative, Accusative and Active*. Niemeyer, Tübingen, 1999.

George Smith
Universität Potsdam
Institut für Germanistik
Postfach 601553
14415 Potsdam
Germany
smithg@rz.uni-potsdam.de
http://www.uni-potsdam.de/u/germanistik/ls_dgs/gs/