A query tool for syntactically annotated corpora*

Laura Kallmeyer
UFRL, Université Paris 7
2, place Jussieu
75251 Paris cedex 05
laura.kallmeyer@linguist.jussieu.fr

Abstract

This paper presents a query tool for syntactically annotated corpora. The query tool is developed to search the Verbmobil treebanks annotated at the University of Tübingen. However, in principle it also can be adapted to other corpora such as the Negra Corpus, the Penn Treebank or the French treebank developed in Paris. The tool uses a query language that allows to search for tokens, syntactic categories, grammatical functions and binary relations of (immediate) dominance and linear precedence between nodes. The overall idea is to extract in an initializing phase the relevant information from the corpus and store it in a relational database. An incoming query is then translated into a corresponding SQL query that is evaluated on the database.

1 Introduction

1.1 Syntactic annotation and linguistic research

With the increasing availability of large amounts of electronic texts, linguists have access to more and more material for empirically based linguistic research. Furthermore, electronic corpora are more and more richly annotated and thereby more and more detailed and structured information contained in the corpora becomes accessable. Currently many corpora are tagged with morphosyntactic categories (part-of-speech) and there are already several syntactically annotated corpora. Examples are the Penn Treebank (Marcus et al., 1994; Bies et al., 1995) annotated at the University of Pennsylvania, the Negra corpus (Brants et al., 1999) developed in Saarbrücken, the Verbmobil treebanks (Hinrichs et al., 2000) annotated in Tübingen and the French treebank annotated in Paris (Abeillé and Clément, 1999).

However, in order to have access to these rich linguistic annotations, adequate query tools are needed.

In the following, an example of a linguistically relevant construction is considered that illustrates how useful access to structural information in a corpus might be.

(1)
über Chomsky habe ich ein Buch
about Chomsky have I a book
gelesen
read
'I have read a book about Chomsky'

Linguists are often concerned with constructions that seem not very natural and where intuitions about grammaticality fail. An example is (1) where we have an accusative object (*ein Buch*) which is positioned between the two verbal elements and whose modifier (the prepositional phrase *über Chomsky*) is topicalized.

Some people claim (1) to be ungrammatical whereas other people are inclined to accept it. In these cases it is very useful to search in an adequate corpus for more natural data showing the same construction (see also (Meurers, 1999) for other examples of the use of corpora for linguistic research).

In order to find structures like (1) in a German corpus, one needs to search for

(a) a prepositional phrase modifying the accusative object and preceding the finite verb (i.e. in the so-called *vorfeld*), and

(b) an accusative object between finite verb and infinite verb forms (i.e. in the so-called *mittelfeld*)

Obviously, two things need to be available in order to enable such a search. On the one hand, one needs a corpus with an annotation that is rich enough to encode the properties

*The work presented here was done as part of a project in SFB 441 "Linguistic Data Structures" at the University of Tübingen.

(a) and (b). On the other hand, one needs a query tool with a query language that allows to express the properties (a) and (b).

Corpora encoding features such as (a) and (b) are for example the Verbmobil treebanks.

1.2 Current query tools

Query tools such as xkwic (Christ, 1994) that allow to search on the tokens and their part-of-speech categories using regular expressions do not allow a direct search on the syntactic annotation structures of the corpus, i.e. a search for specific relations between nodes in the annotation such as dominance or linear precedence. Therefore many queries a linguist would like to ask using a syntactically annotated corpus either cannnot be expressed in a regular expression based language or at least cannot be expressed in an intuitive way.

Even more recent query languages as SgmlQL (Le Maitre et al., 1998) and XML-QL (Deutsch et al., 1999) that refer to the SGML or XML annotation of a corpus are in general not adequate for syntactically annotated corpora: if the annotations are trees and the nesting of SGML/XML elements encodes the tree structure, such query languages work nicely. With regular path expressions as supported by XML-QL, it is possible to search not only for parent but also for dominance relations. However, in order to deal with discontinuous constituents, most syntactically annotated corpora do not contain trees but slightly different data structures. The Penn Treebank for example consists of trees with an additional coindexation relation, Negra allows crossing branches and in Verbmobil, an element (a tree-like structure) in the corpus might contain completely disconnected nodes. In order to express these annotations in XML, one has to encode for example each node and each edge as a single element as in (Mengel and Lezius, 2000). But then a query for a dominance relation can no longer be formulated with a regular path expression.

In this paper, I propose a query tool that allows to search for parent, dominance and linear precedence relations even in corpora annotated with structures slightly different from trees.

2 The Verbmobil treebanks

The German Verbmobil corpus (Stegmann et al., 1998; Hinrichs et al., 2000) is a treebank annotated at the University of Tübingen that contains approx. 38.000 trees (or rather tree-like annotation structures since, as already mentioned, the structures are not always trees). The corpus consists of spoken texts restricted to the domain of arrangement of business appointments.

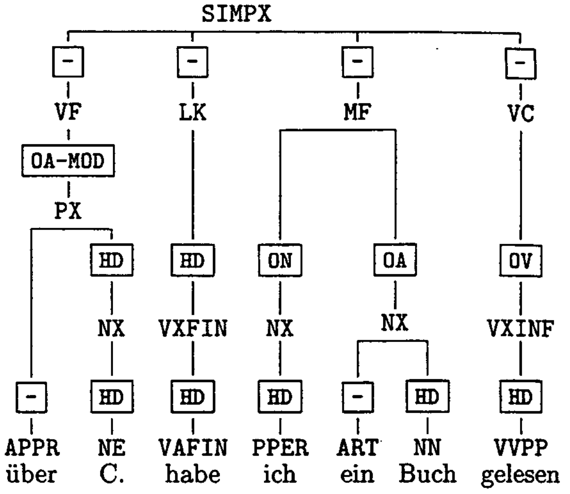

Figure 1: Annotation of (1) in Verbmobil format

The Verbmobil corpus is part-of-speech tagged using the Stuttgart Tübingen tagset (STTS) described in (Schiller et al., 1995).

One of the design decisions in Verbmobil was that for the purpose of reusability of the treebank, the annotation scheme should not reflect a commitment to a particular syntactic theory. Therefore a surface-oriented annotation scheme was adopted that is inspired by the notion of topological fields in the sense of (Höhle, 1985). The discontinuous positioning of the verbal elements in verb-first and verb-second sentences (as in (1) for example) is the traditional reason to structure the German sentence by means of topological fields: The verbal elements have the categories LK (*linke Klammer*) and VC (*verbal complex*), and roughly everything preceding the LK forms the 'vorfeld' VF, everything between LK and VC forms the 'mittelfeld' MF and the 'nachfeld' NF follows the verbal complex.

The Verbmobil corpus is annotated with syntactic categories as node labels, grammatical functions as edge labels and dependency relations. The syntactic categories are based on traditional phrase structure and on the theory of topological fields. In contrast to Negra or Penn Treebank, there are neither crossing branches nor empty categories. Instead, dependency relations are expressed within the grammatical functions (e.g. OA-MOD for a constituent modifying the accusative object).

A sample annotation conformant to the Verbmobil annotation scheme is the annotation of (1) shown in Fig. 1. (The elements set in boxes are edge labels.)

In order to search for structures as in Fig. 1, one needs to search for trees containing a node n_1 with label PX and grammatical function OA-MOD, a node n_2 with label VF that dominates n_1, a node n_3 with label MF and a node n_4 with label NX and grammatical function OA that is immediately dominated by n_3.

Evaluating a query for structures as in Fig. 1 on the Verbmobil corpus gives results such as (2) that sound much more natural than the constructed example (1).

(2) tja , über Flugverbindungen habe ich
about flight connections have I
leider keine Information .
unfortunately no information .
'unfortunately I have no information about flight connections.'

This example illustrates the usefulness of syntactic annotations for linguistic research and it shows the need of query languages and query tools that allow access to these annotations.

3 The query language

3.1 Syntax

As query language for the German Verbmobil corpus, a first order logic without quantification is chosen where variables are interpreted as existentially quantified. Negation is only allowed for atomic formula. It seems that even this very simple logic already gives a high degree of expressive power with respect to the queries linguists are interested in (see for example (Kallmeyer, 2000) for theoretical investigations of query languages). However, it might be that at a later stage the query language will be extended.

Let C (the node labels, i.e. syntactic categories and part-of-speech categories), E (the edge labels, i.e. grammatical functions) and T (the terminals, i.e. tokens) be pairwise disjoint finite sets. `>`, `>>`, `..` are constants for the binary relations immediate dominance (parent relation), dominance (reflexive transitive closure of immediate dominance) and linear precedence. The set $\mathbb{N}$ of natural numbers is used as variables. Further, `&`, `|`, `!` are logical connectives (conjunction, disjunction, and negation).

Definition 1 (**(C, E, T)-queries**)
(C, E, T)-queries are inductively defined:

(a) for all i$\in \mathbb{N}$, t$\in T$:
token(i)=t *and* token(i)!=t *are queries,*

(b) for all i$\in \mathbb{N}$, c$\in C$:
cat(i)=c *and* cat(i)!=c *are queries,*

(c) for all i$\in \mathbb{N}$, e$\in E$:
fct(i)=e *and* fct(i)!=e *are queries,*

(d) for all i, j$\in \mathbb{N}$:
i > j *and* i !> j *are queries,*
i >> j *and* i !>> j *are queries,*
i .. j *and* i !.. j *are queries,*

(e) for all queries q_1, q_2*:*
q_1 & q_2 *and* (q_1 | q_2) *are queries.*

Of course, when adapting this language to another corpus, depending on the specific annotation scheme, other unary or binary predicates might be added to the query language. This does not change the complexity of the query language in general.

However, it is also possible that at a later point negation needs to be allowed in a general way or that quantification needs to be added to the query language for linguistic reasons. Such modifications would affect the complexity of the language and the performance of the tool. Therefore the decision was taken to keep the language as simple as possible in the beginning.

3.2 Intended models

In the case of the German Verbmobil corpus, the data structures are not trees, since structures as in Fig. 2, which shows the annotation of the long-distance wh-movement in (3), can occur. The structure in Fig. 2 does not have a unique root node, and the two nodes with label SIMPX have neither a dominance nor a linear precedence relation.

(3) wen glaubst du liebt Maria
whom believe you loves Maria
'whom do you believe Maria loves'

Therefore, the models of our queries are defined as more general structures than finite trees.

A model is a tuple $(\mathcal{U}, \mathcal{P}, \mathcal{D}, \mathcal{L}, \mu, \eta, \sigma)$ where $\mathcal{U}$ is the set of nodes, $\mathcal{P}, \mathcal{D}$ and $\mathcal{L}$ are the

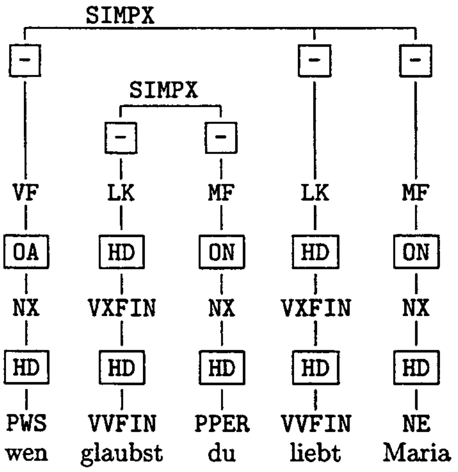

Figure 2: Annotation of (3) in Verbmobil format

binary relations immediate dominance (parent), dominance and linear precedence, μ is a function assigning syntactic categories or part-of-speech tags to nodes, η is a function mapping edges to grammatical functions, and σ assigns tokens to the leaves (i.e. the nodes that do not dominate any other node).

Definition 2 (Query model)
Let C, E and T be disjoint alphabets.
$(\mathcal{U}, \mathcal{P}, \mathcal{D}, \mathcal{L}, \mu, \eta, \sigma)$ is a query model with categories C, edge labels E and terminals T iff

1. *$\mathcal{U}$ is a finite set with $\mathcal{U} \cap (C \cup E \cup T) = \emptyset$, the set of nodes.*
2. *$\mathcal{P}, \mathcal{L}, \mathcal{D} \in \mathcal{U} \times \mathcal{U}$, such that:*
   (a) *$\mathcal{P}$ is irreflexive, and for all $x \in \mathcal{U}$ there is at most one $v \in \mathcal{U}$ with $\langle v, x \rangle \in \mathcal{P}$.*
   (b) *$\mathcal{D}$ is the reflexive transitive closure of $\mathcal{P}$, and $\mathcal{D}$ is antisymmetric.*
   (c) *$\mathcal{L}$ is transitive.*
   (d) *for all $x, y \in \mathcal{U}$: if $\langle x, y \rangle \in \mathcal{L}$, then $\langle x, y \rangle \notin \mathcal{D}$ and $\langle y, x \rangle \notin \mathcal{D}$.*
   (e) *for all $x, y \in \mathcal{U}$: $\langle x, y \rangle \in \mathcal{L}$ iff for all $z, w \in \mathcal{U}$ with $\langle x, z \rangle, \langle y, w \rangle \in \mathcal{D}$, $\langle z, w \rangle \in \mathcal{L}$ holds.*
   (f) *for all $x, y, z \in \mathcal{U}$: if $\langle x, y \rangle, \langle x, z \rangle \in \mathcal{D}$, then either $\langle x, z \rangle \in \mathcal{D}$ or $\langle z, x \rangle \in \mathcal{D}$ or $\langle x, z \rangle \in \mathcal{L}$ or $\langle z, x \rangle \in \mathcal{L}$.*
3. *$\mu : \mathcal{U} \to C$ is a total function.*
4. *$\eta : \mathcal{P} \to E$ is a total function.*
5. *$\sigma : \{u \in \mathcal{U} \mid$ there is no u' with $\langle u, u' \rangle \in \mathcal{P}\} \to T$ is a total function.*

With (b), (c) and (d), $\mathcal{L}$ is also irreflexive and antisymmetric.

In contrast to finite trees, our query models do not necessarily have a unique root node, i.e. a node that dominates all other nodes. Consequently, the so-called *exhaustiveness* property does not hold since two nodes in a query model might be completely disconnected. In other words, it does not hold in general that $\langle x, y \rangle \in \mathcal{D}$ or $\langle y, x \rangle \in \mathcal{D}$ or $\langle x, y \rangle \in \mathcal{L}$ or $\langle y, x \rangle \in \mathcal{L}$ for all $x, y \in \mathcal{U}$. This holds only for nodes $x, y \in \mathcal{U}$ where a node z exists that dominates x and y.

3.3 Semantics

Satisfiability of a query q by a query model M is defined in the classical model-theoretic way with respect to an injective assignment g mapping node variables to nodes in the query model.

Definition 3 (Satisfiability)
Let $M = (\mathcal{U}, \mathcal{P}, \mathcal{D}, \mathcal{L}, \mu, \eta, \sigma)$ be a query model and let $g : \mathbb{N} \to \mathcal{U}$ be an injective function.
For all i$\in \mathbb{N}$, t$\in T$, c$\in C$, e$\in E$:

- $M \models_g$ `token(i)=t` *iff* $\sigma(g(\texttt{i})) = \texttt{t}$.
- $M \models_g$ `token(i)!=t` *iff* $\sigma(g(\texttt{i})) \neq \texttt{t}$.
- $M \models_g$ `cat(i)=c` *iff* $\mu(g(\texttt{i})) = \texttt{c}$.
- $M \models_g$ `cat(i)!=c` *iff* $\mu(g(\texttt{i})) \neq \texttt{c}$.
- $M \models_g$ `fct(i)=e` *iff there is a* $u \in \mathcal{U}$ *with* $\langle u, g(\texttt{i}) \rangle \in \mathcal{P}$ *and* $\eta(\langle u, g(\texttt{i}) \rangle) = \texttt{e}$.
- $M \models_g$ `fct(i)!=e` *iff there is no* $u \in \mathcal{U}$ *with* $\langle u, g(\texttt{i}) \rangle \in \mathcal{P}$ *and* $\eta(\langle u, g(\texttt{i}) \rangle) = \texttt{e}$.

For all i, j$\in \mathbb{N}$:

- $M \models_g$ `i > j` *iff* $\langle g(\texttt{i}), g(\texttt{j}) \rangle \in \mathcal{P}$ *(i.e.* $g(\texttt{i})$ *immediately dominates* $g(\texttt{j})$*)*
- $M \models_g$ `i !> j` *iff* $\langle g(\texttt{i}), g(\texttt{j}) \rangle \notin \mathcal{P}$
- $M \models_g$ `i >> j` *iff* $\langle g(\texttt{i}), g(\texttt{j}) \rangle \in \mathcal{D}$ *(i.e.* $g(\texttt{i})$ *dominates* $g(\texttt{j})$*)*
- $M \models_g$ `i !>> j` *iff* $\langle g(\texttt{i}), g(\texttt{j}) \rangle \notin \mathcal{D}$
- $M \models_g$ `i .. j` *iff* $\langle g(\texttt{i}), g(\texttt{j}) \rangle \in \mathcal{L}$ *(i.e.* $g(\texttt{i})$ *is left of* $g(\texttt{j})$*)*
- $M \models_g$ `i !.. j` *iff* $\langle g(\texttt{i}), g(\texttt{j}) \rangle \notin \mathcal{L}$

For all queries q_1, q_2:

- $M \models_g q_1$ `&` q_2 *iff* $M \models_g q_1$ *and* $M \models_g q_2$
- $M \models_g$ `(`q_1 `|` q_2`)` *iff* $M \models_g q_1$ *or* $M \models_g q_2$.

Note that the condition that g needs to be injective means that different variables are considered to refer to different nodes. In this respect, Def. 3 differs from traditional model-theoretic semantics.

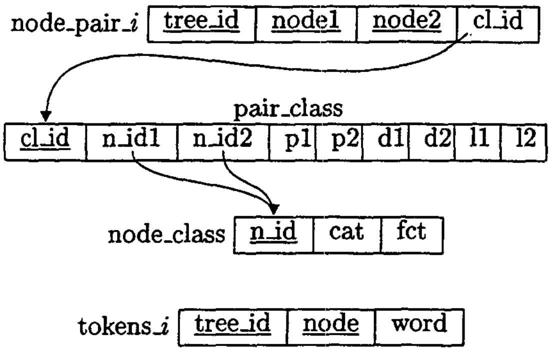

Figure 3: The relational database schema

As an example, consider the query for structures as in (1) that is shown in (4). The structure in Fig. 1 is a query model satisfying (4).

```
(4) cat(1)=PX & fct(1)=OA-MOD
    & cat(2)=VF & 2>>1
    & cat(3)=MF & cat(4)=NX
    & fct(4)=OA & 3>>4
```

4 Storing the corpus in a database

As already mentioned, the general idea of the query tool is to store the information one wants to search for in a relational database and then to translate an expression in the query language presented in the previous section into an SQL expression that is evaluated on the database. The first part is performed by an initializing component and needs to be done only once per corpus, usually by the corpus administrator. The second part, i.e. the querying of the corpus, is performed by a query component.

The tool is implemented in Java with Java Database Connectivity (JDBC) as interface and mysql as database management system.

4.1 The relational database schema

The German Verbmobil corpus consist of several subcorpora. In the relational database there are two global tables, node_class and pair_class. Besides these, for each of the subcorpora identified by i there are tables tokens_i and node_pair_i. The database schema is shown in Fig. 3. The arrows represent foreign keys. The column cl_id in the table node_pair_i, for example, is a foreign key referring to the column cl_id in the table pair_class. This means that each entry for cl_id in node_pair_i uniquely refers to one entry for cl_id in pair_class.

The content of the tables is as follows:

- **node_class** contains node classes characterized by category (node label) and grammatical function (edge label between the node and its mother). Each node class has a unique identifier, namely the column n_id.
- **pair_class** contains classes of node pairs characterized by the two node classes and the parent, dominance and linear precedence relation between the two nodes. The columns p1, p2, d1, d2, l1 and l2 stand for binary relations and have values 1 or 0 depending on whether the relation holds or not. p1 signifies immediate dominance of the first node over the second, p2 immediate dominance of the second over the first, d1 dominance of the first over the second, etc. Each node pair class has a unique identifier, namely its cl_id.
- **tokens_i** contains all leaves from subcorpus i with their tokens (word).
- **node_pair_i** contains all node pairs from subcorpus i with their pair class. Of course, only pairs of nodes belonging to one single annotation structure are stored.

4.2 Initializing the database

The storage of the corpus in the database is done by an initializing component. This component extracts information from the structures in export format (the format used for the German Verbmobil corpus) and stores them in the database. The export format explicitly encodes tokens, categories and edge labels, linear precedence between leaves and the parent (immediate dominance) relation. Dominance and linear precedence in general however need to be precompiled.

First the dominance relation is computed simply as reflexive transitive closure of the parent relation.

Linear precedence on the leaves can be immediately extracted from the export format. When computing linear precedence for internal nodes, the specific properties of the data structures in Verbmobil (see Section 3) must be taken into account. Unlike in finite trees, for two nodes u_1, u_2, the fact that u_1 dominates some x and u_2 dominates some y and x is left of y is not sufficient to decide that u_1 is left of u_2. Instead (see axiom (e) in Def. 2) the following holds for two nodes u_1, u_2: u_1 is left of u_2 iff for all x, y dominated by u_1, u_2 respectively: x is left of y.

In general, the database schema itself does

```
#BOS 24 25 898511955 1
scheinbar     ADV       --   HD   500
nicht         PTKNEG    --   HD   501
beides        PIS       --   HD   502
zusammen      ADV       --   HD   503
.             $.        --   --   0
#500          ADVX      --   -    505
#501          ADVX      --   -    505
#502          NX        --   HD   504
#503          ADVX      --   -    504
#504          NX        --   HD   505
#505          NX        --   --   0
#EOS 24
```

Corresponding structure:

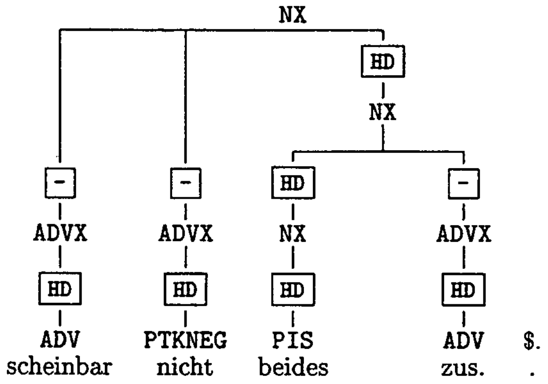

Figure 4: Export format of sentence 24 in cd20 and corresponding structure

tokens_20

tree_id	node	word
24	0	scheinbar
24	1	nicht
24	2	beides
24	3	zusammen
24	4	.

node_pair_20

tree_id	node1	node2	cl_id
24	0	1	1459
24	0	2	2608
24	0	3	120
		...	
24	9	10	1327
		...	

pair_class

cl_id	n_id1	n_id2	p1	p2	d1	d2	l1	l2
			...					
120	13	13	0	0	0	0	1	0
			...					
1327	24	25	0	1	0	1	0	0
			...					

node_class

n_id	cat	fct
	...	
13	ADV	HD
	...	
24	NX	HD
25	NX	–
	...	

Figure 5: Sentence 24 in the database

not reflect the concrete properties of the query model, in particular the properties of the binary relations are not part of the database schema, e.g. considering only the database, the dominance and linear precedence relations are not necessarily transitive. Therefore, the query tool can be easily adapted to other data structures, for example to feature structures with reentrancy as annotations. In this case, a modification of the part of the initializing component that computes the binary relations would be sufficient.

As an example, consider how sentence 24 in the subcorpus cd20 (identifier 20) is stored in the database. This sentence was chosen for the simple reason that it is not too long but contains enough nodes to provide a useful example. Besides this, its construction and its tokens are not of any interest here.

Fig. 4 shows the sentence in its export format, i.e. the way it originally occurs in the corpus, together with a picture of the corresponding structure. Parts of the tables in the database concerning sentence 24 are shown in Fig. 5. Each line in the export format corresponds to one node. The nodes are assigned numbers 0, 1, ... in the order of the lines in the export format. The nodes with tokens (i.e. that are leaves) are inserted into the table tokens_20. Furthermore, each pair of nodes occurring in sentence 24 is inserted into the table node_pair_20 together with its pair class. Both orders of a pair are stored.[1] The pair classes and node classes belonging to a pair can be found in the two global tables. Consider for example the nodes 9 and 10 in sentence 24 (the node labelled NX that dominates *beides zusammen* and the topmost node with label NX). The cl_id of this pair is 1327.

[1] In a previous version just one order was stored but it turned out that for some queries this causes an exponential time complexity depending on the number of variables occurring in the query. This problem is avoided storing both orders of a node pair.

The corresponding entry in pair_class tells us that the second node is the mother of the first, that the second dominates the first, and that there is no linear precedence relation between the two nodes. Furthermore, the node classes identified by n_id1 and n_id2 are such that the first node has label NX and grammatical function HD whereas the second has label NX and no grammatical function.

4.3 The size of the database

So far, in order to test the tool, approximately one quarter of the German Verbmobil corpus is stored in the database, namely the following subcorpora:

id	sub-corpus	trees	tokens	pairs
15	cd15	1567	15474	1326416
20	cd20	2151	21069	1941056
21	cd21	2416	22360	1761082
22	cd22	1723	16587	1229324
24	cd24	2255	22763	2129548

The table pair_class has 23024 entries and node_class has 213 entries. The following table shows the current size of the files:

Table	data file (KB)	index file (KB)
node_class	1	10
pair_class	585	1500
tokens_15	303	293
tokens_20	413	403
tokens_21	439	423
tokens_22	325	311
tokens_24	446	430
node_pair_15	9067	72556
node_pair_20	13269	106377
node_pair_21	12039	96383
node_pair_22	8404	67153
node_pair_24	14557	116694

5 Searching the corpus

In order to search the corpus, one needs of course to know the specific properties of the annotation scheme. These are described in the STTS guidelines (Schiller et al., 1995) and the Verbmobil stylebook (Stegmann et al., 1998) that must be both available to any user of the query tool.

Currently, the query component does not yet process all possible expressions in the query language. In particular, it does not allow disjunctions and it does not allow to query for tokens. Other atomic queries combined with with negations and conjunctions are possible. In particular, complex syntactic structures involving category and edge labels and binary relations can be searched. The query component will be completed very soon to process all queries defined in Section 3.

The query component takes an expression in the query language as input and translates this into a corresponding SQL expression, which is then passed to the database.

As an example, consider again the query (4) repeated here as (5):

```
(5) cat(1)=PX & fct(1)=OA-MOD &
    cat(2)=VF & 2>>1 & cat(3)=MF &
    cat(4)=NX & fct(4)=OA & 3>>4
```

For query (5) as input performed on the subcorpus cd20, the query component produces the following SQL query:

```
SELECT DISTINCT np1.tree_id
FROM
node_class AS nc1, node_class AS nc2,
node_class AS nc3, node_class AS nc4,
node_pair_20 AS np1, pair_class AS pc1,
node_pair_20 AS np2, pair_class AS pc2
WHERE
nc1.cat='PX' AND nc1.fct='OA-MOD'
AND nc2.cat='VF' AND nc3.cat='MF'
AND nc4.cat='NX' AND nc4.fct='OA'
AND pc1.n_id1=nc2.n_id
AND pc1.n_id2=nc1.n_id AND pc1.d1=1
AND pc2.n_id1=nc3.n_id
AND pc2.n_id2=nc4.n_id AND pc2.d1=1
AND np1.cl_id=pc1.cl_id
AND np1.tree_id=np2.tree_id
AND np2.cl_id=pc2.cl_id;
```

As a second example consider the search for long distance wh-movements as in (3). The annotation of (3) using the Verbmobil annotation scheme was shown in Fig. 2. Such structures might be characterized by the following properties: there is an interrogative pronoun (part-of-speech tag PWS for substituting interrogative pronoun) that is part of a simplex clause and there is another simplex clause containing a finite verb such that the two simplex clauses are not connected and the pronoun precedes the finite verb. This leads to the query (6):

```
(6) cat(1)=PWS & cat(2)=SIMPX & 2>>1
    & cat(3)=SIMPX & cat(4)=VVFIN
    & 2!>>3 & 3!>>2 & 2!..3 & 3!..2
    & 1..4 & 3>>4
```

Performed on cd20, (6) as input leads to the following SQL query:

```
SELECT DISTINCT np1.tree_id
FROM
node_class AS nc1, node_class AS nc2,
node_class AS nc3, node_class AS nc4,
node_pair_20 AS np1,
pair_class AS pc1,
node_pair_20 AS np2,
pair_class AS pc2,
node_pair_20 AS np3,
pair_class AS pc3,
node_pair_20 AS np4,
pair_class AS pc4
WHERE
nc1.cat='PWS' AND nc2.cat='SIMPX'
AND nc3.cat='SIMPX'
AND nc4.cat='VVFIN'
AND pc1.n_id1=nc2.n_id
AND pc1.n_id2=nc1.n_id AND pc1.d1=1
AND pc2.n_id1=nc2.n_id
AND pc2.n_id2=nc3.n_id
AND pc2.d1=0 AND pc2.d2=0
AND pc2.l1=0 AND pc2.l2=0
AND pc3.n_id1=nc1.n_id
AND pc3.n_id2=nc4.n_id AND pc3.l1=1
AND pc4.n_id1=nc3.n_id
AND pc4.n_id2=nc4.n_id AND pc4.d1=1
AND np1.cl_id=pc1.cl_id
AND np1.node1=np2.node1
AND np1.node2=np3.node1
AND np1.tree_id=np2.tree_id
AND np2.cl_id=pc2.cl_id
AND np2.node2=np4.node1
AND np2.tree_id=np3.tree_id
AND np3.cl_id=pc3.cl_id
AND np3.node2=np4.node2
AND np3.tree_id=np4.tree_id
AND np4.cl_id=pc4.cl_id;
```

Currently the database and the tool are running on a Pentium II PC 400MHz 128MB under Linux. On this machine, example (5) takes 1.46 sec to be answered by mysql, and example (6) takes 6.43 sec to be answered. This shows that although the queries, in particular the last one, are quite complex and involve many intermediate results, the performance of the system is quite efficient.

The performance of course depends crucially on the size of intermediate results. In cases where more than one node pair is searched for (as in the two examples above) the order of the pairs is important since the result set of the first pair restricts the second pair. In (5) for example, first a node pair with a `PX` with function `OA-MOD` dominated by a `VF` is searched for. Afterwards, the search for the `NX` with function `OA` in the `MF` is restricted to those trees that were found when searching for the first pair. Obviously, the first pair is much more restrictive than the second. If the order is reversed, the query takes much more time to process. Currently the ordering of the pairs needs to be done by the user, i.e. depends on the incoming query. However, we plan to implement at least partly an ordering of the binary conjuncts in the query depending on the frequency of the syntactic categories and grammatical functions involved in the pairs.

The obvious advantage of using a relational database to store the corpus is that some parts of the work are taken over by the database management system such as the search of the corpus. Furthermore, and this is crucial, the indexing functionalities of the database management system can be used to increase the performance of the tool, e.g. indexes are put on cl_id in node_pair_*i* and on n_id1 and n_id2 in pair_class.

6 Conclusion and future work

In this paper, I have presented a query tool for syntactically annotated corpora that is developed for the German Verbmobil treebank annotated at the University of Tübingen. The key idea is to extract in an initializing phase the information one wants to search for from the corpus and to store it in a relational database. The search itself is done by translating an input query that is an expression in a simple quantifier free first order logic into an SQL query that is then passed to the database system.

An obvious advantage of this architecture is that a considerable amount of work is taken over by the database management system and therefore needs not to be implemented. Furthermore, the mysql indexing functionalities can be used to directly affect the performance of the search.

The query tool is work in progress, and I briefly want to point out some of the things that still need to be done. First, the set of queries the tool can process needs to be extended to all queries allowed in the query language. This will be done very soon. Another task for the near future is, as mentioned in the previous section, to add an ordering mechanism on binary conjuncts in order to ensure that the more restrictive node pairs are searched for first. Further, the design of a graphical user-interface to enter the queries is planned, allowing to specify queries by drawing partial trees instead of typing in the expressions in the query language. Finally, we also want to implement a web-based user-interface for the query tool.

Besides these tasks that all concern the current query tool for the German Verbmobil corpus, a more general issue to persue in the future is to adapt the tool to other corpora. In some cases, this implies a modification of the way binary relations are precompiled, and in some other cases this would even lead to a modification of the query language and the database schema, namely in those cases where other binary relations are needed, e.g. the coindexation relation in the case of the Penn Treebank.

Acknowledgments

For fruitful discussions I would like to thank Oliver Plaehn and Ilona Steiner. Furthermore, I am grateful to three anonymous reviewers for their valuable comments.

References

Anne Abeillé and Lionel Clément. 1999. A tagged reference Corpus for French. In *Proceedings of EACL-LINC*, Bergen.

Ann Bies, Mark Ferguson, Karen Katz, and Robert MacIntyre. 1995. Bracketing Guidelines for Treebank II Style Penn Treebank Project. University of Pennsylvania.

Thorsten Brants, Wojciech Skut, and Hans Uszkoreit. 1999. Syntactic Annotation of a German Newspaper Corpus. In *Journées ATALA, 18–19 juin 1999, Corpus annotés pour la syntaxe*, pages 69–76, Paris.

Oliver Christ. 1994. A modular and flexible architecture for an integrated corpus query system. In *Proceedings of COMPLEX'94*.

Alin Deutsch, Mary Fernandez, Daniela Florescu, Alon Levy, and Dan Suciu. 1999. A Query Language for XML. In *Proceedings of the International World Wide Web Conference (WWW)*, volume 31, pages 1155–1169.

Erhard W. Hinrichs, Julia Bartels, Yasuhiro Kawata, Valia Kordoni, and Heike Telljohann. 2000. The VERBMOBIL Treebanks. In *Proceedings of KONVENS 2000*, October. To appear.

Tilman Höhle. 1985. Der Begriff 'Mittelfeld'. Anmerkungen über die Theorie der topologischen Felder. In A. Schöne, editor, *Kontroversen alte und neue. Akten des 7. Internationalen Germanistenkongresses Göttingen*, pages 329–340.

Laura Kallmeyer. 2000. On the Complexity of Queries for Structurally Annotated Linguistic Data. In *Proceedings of ACIDCA'2000*, pages 105–110, March.

Jacques Le Maitre, Elisabeth Murisasco, and Monique Rolbert. 1998. From Annotated Corpora to Databases: the SgmlQL Language. In John Nerbonne, editor, *Linguistic databases*. CSLI.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating Predicate Argument Structure. In *ARPA '94*.

Andreas Mengel and Wolfgang Lezius. 2000. An XML-based encoding format for syntactically annotated corpora. In *Proceedings of LREC 2000*.

Detmar Meurers. 1999. Von partiellen Konstituenten, erstaunlichen Passiven und verwirrten Franken. zur Verwendung von Korpora für die theoretische Linguistik. Handout at the DGfS Jahrestagung, February.

A. Schiller, S. Teufel, and C. Thielen. 1995. Guidelines für das Tagging deutscher Textcorpora mit STTS. Manuskript Universität Stuttgart und Universität Tübingen.

Rosemary Stegmann, Heike Schulz, and Erhard W. Hinrichs. 1998. Stylebook for the German Treebank in VERBMOBIL. Eberhard-Karls Universität Tübingen.