Netgraph – Making Searching in Treebanks Easy

Jiří Mírovský
Charles University in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
Malostranské nám. 25, 118 00 Prague 1, Czech Republic
mirovsky@ufal.mff.cuni.cz

1 Introduction

Searching in a linguistically annotated treebank is a principal task that requires a sophisticated tool. Netgraph has been designed to perform the searching with maximum comfort and minimum requirements on its users. Although it has been developed primarily for the Prague Dependency Treebank 2.0 (Hajič et al. 2006), it can be used with other treebanks too, both dependency and constituent-structure types, as long as the treebank is transformed to a suitable format.

In this paper, we present Netgraph query language and on many examples show how it can be used to search for frequent linguistic phenomena.

In *section 1* (after this introduction) we extremely briefly describe the Prague Dependency Treebank 2.0, just to make the examples in the subsequent text more understandable. In the next subsection we mention the history of Netgraph and its properties as a tool.

In *section 2* we offer an introduction to the query language of Netgraph along with the idea of meta-attributes and what they are good for, and present linguistically motivated examples of queries in the Prague Dependency Treebank.

Finally, in *section 3* we offer some concluding remarks.

1.1 Prague Dependency Treebank 2.0

The Prague Dependency Treebank 2.0 (PDT 2.0, see Hajič et al. 2006, Hajič 2004) is a manually annotated corpus of Czech. It is a sequel to the Prague Dependency Treebank 1.0 (PDT 1.0, see Hajič et al. 2001a, Hajič et al. 2001b).

The texts in PDT 2.0 are annotated on three layers - the morphological layer, the analytical layer and the tectogrammatical layer. The corpus size is almost 2 million tokens (115 thousand sentences), although "only" 0.8 million tokens (49 thousand sentences) are annotated on all three layers. By 'tokens' we mean word forms, including numbers and punctuation marks.

On the morphological layer (Hana et al. 2005), each token of every sentence is annotated with a lemma (attribute `m/lemma`), keeping the base form of the token, and a tag (attribute `m/tag`), keeping its morphological information. Sentence boundaries are annotated here, too.

The analytical layer roughly corresponds to the surface syntax of the sentence; the annotation is a single-rooted dependency tree with labeled nodes (Hajič et al. 1997, Hajič 1998). Attribute `afun` describes the type of dependency between a dependent node and its governor. The nodes on the analytical layer (except for technical roots of the trees) also correspond 1:1 to the tokens of the sentences. The order of the nodes from left to right corresponds exactly to the surface order of tokens in the sentence (attribute `ord`). Non-projective constructions (that are quite frequent both in Czech (Hajičová et al. 2004) and in some other languages (see Havelka 2007)) are allowed.

The tectogrammatical layer captures the linguistic meaning of the sentence in its context. Again, the annotation is a dependency tree with labeled nodes (see Hajičová 1998). The correspondence of the nodes to the lower layers is more complex here. It is often not 1:1, it can be both 1:N and N:1. It was shown in detail in Mírovský (2006) how Netgraph deals with this issue.

Attribute `functor` describes the dependency between a dependent node and its governor. A tec-

togrammatical lemma (attribute t_lemma) is assigned to every node. Grammatemes, which keep additional annotation, are rendered as a set of 16 attributes grouped by the "prefix" gram (e.g. gram/verbmod for verbal modality).

Topic and focus (Hajičová et al. 1998) are marked (attribute tfa), together with so-called deep word order reflected by the order of nodes in the annotation (attribute deepord).

Coreference relations between nodes of certain category types are captured (Kučová et al. 2003), distinguishing also the type of the relation (textual or grammatical). Each node has an identifier (attribute id) that is unique throughout the whole corpus. Attributes coref_text.rf and coref_gram.rf contain ids of coreferential nodes of the respective types.

1.2 Netgraph as a Tool

The development of Netgraph started in 1998 as a topic of Ondruška's Master's thesis (Ondruška 1998), and has been proceeding along with the ongoing annotations of the Prague Dependency Treebank 1.0 and later the Prague Dependency Treebank 2.0. Now it is a fully functional tool for complex searching in PDT 2.0 and other treebanks.

Netgraph is a client-server application that allows multiple users to search the treebank on-line and simultaneously through the Internet. The server (written in C) searches the treebank, which is located at the same computer or local network. The client (written in Java2) serves as a very comfortable graphical user interface and can be located at any node in the Internet. The client exists in two forms: as an applet and as a stand-alone application. The applet version can be run from Netgraph home page and searches in PDT 2.0. The stand-alone version can be downloaded from the same page and can connect anonymously to PDT 2.0 server. More information can be found on Netgraph home page (http://quest.ms.mff.cuni.cz/netgraph).

The client sends user queries to the server and receives results from it. Both the server and the client also can, of course, reside at the same computer. Authentication by the means of login names and passwords is provided. Users can have various access permissions.

A detailed description of the inner architecture of Netgraph and of the communication between the server and the client was given in Mírovský, Ondruška and Průša (2002).

Netgraph server requires the treebank in FS format, encoded in UTF-8. A formal description of the format can be found in Hajič et al. 2001a. Netgraph query language, presented in the next section, is an extension of FS format.

2 Netgraph Query Language

In this section we give an introduction to Netgraph query language. We show on a series of examples how some frequent linguistic phenomena can be searched for.

2.1 The Query Is a Tree

The query in Netgraph is a tree that forms a subtree in the result trees. The treebank is searched tree by tree and whenever the query is found as a subtree of a tree (we say the query and the tree match), the tree becomes part of the result. The result is displayed tree by tree on demand. The query can also consist of several trees joined either by AND or OR relation. In that case, all the query trees at the same time (or at least one of the query trees, respectively) are required to match the result tree.

The query has both a textual form and a graphical form. In the following text, we will use its textual form for simple queries and its graphical form (or both forms) for more advanced queries.

The syntax of the language is very simple. In the textual form, square brackets enclose a node, attributes (pairs name=value) are separated by a comma, quotation marks enclose a regular expression in a value. Parentheses enclose a subtree of a node, brothers are separated by a comma. In multiple-tree queries, each tree is on a new line and the first line contains only a single AND or OR. Alternative values of an attribute, as well as alternative nodes, are separated by a vertical bar. It almost completes the description of the syntax, only one thing (references) will be added in the following subsection.

The simplest possible query (and probably of little interest on itself) is a simple node without any evaluation: []. It matches all nodes of all trees in the treebank, each tree as many times as how many nodes there are in the tree. Nevertheless, we may add conditions on its attributes, optionally using regular expressions in values of the attributes. Thus

we may search e.g. for all nodes that are Subjects and nouns but not in first case:

```
[afun=Sb, m/tag="N...[^1].*"].
```

We may notice here that regular expressions allow the first (very basic) type of negation in queries.

More interesting queries usually consist of several nodes, forming a tree structure. The following example query searches for trees containing a Predicate that directly governs a Subject and an Object:

```
[afun=Pred]([afun=Sb],[afun=Obj]).
```

Please note that there is no condition in the query on the order of the Subject and the Object, nor on their left-right position to their father. It does not prevent other nodes to be directly governed by the Predicate either.

2.2 Meta-Attributes

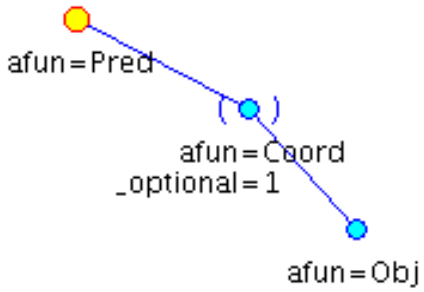
This simple query language, described briefly in only a few examples, is quite useful but not powerful enough. There is no possibility to set a real negation, no way of restricting the position of the query in the result tree or the size of the result tree, nor the order of nodes can be controlled. To allow these and other things, meta-attributes have been added to the query system.

Meta-attributes are not present in the corpus but they pretend to be ordinary attributes and the user uses them the same way like normal attributes. Their names start with an underscore. There are eleven meta-attributes, each adding some power to the query language, enhancing its semantics, while keeping the syntax of the language on the same simple level. We present several of the meta-attributes in this subsection, some others will be presented in the subsequent section, when they are needed. A detailed description of the principal meta-attributes was given in Mírovský (2006).

Coordination is a frequent phenomenon in languages. In PDT (and in most other treebanks, too) it is represented by a coordinating node. To be able to skip (and effectively ignore) the coordination in the queries, we have introduced the meta-attribute `_optional` that marks an optional node. The node then may but does not have to be in the result. If we are interested, for example, in Predicates governing Objects, we can get both cases (with coordination and without it) in one query using this meta-attribute:

```
[afun=Pred]([afun=Coord,_optional=1]([afun=Obj])).
```

The Coordination becomes optional. If there is a node between the Predicate and its Object in the result tree, it has to be the Coordination. But the Object may also be a direct son of the Predicate, omitting the optional Coordination. The picture demonstrates that the graphical representation of the query is much more comprehensible than its textual version:

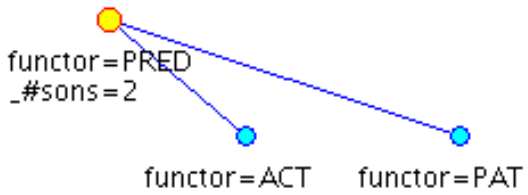


There is a group of meta-attributes of rather technical nature, which allow setting a position of the query in the result tree, restricting the size of the result tree or its part, and restricting number of direct sons of a node. Meta attribute `_depth` controls the distance of a node from the root (useful when searching for a phenomenon in subordinated clauses, for example), `_#descendants` controls number of nodes in the subtree of a node (useful e.g. when searching for „nice“ small examples of something), `_#sons` controls number of (direct) sons of a node.

Controlling number of direct sons (mainly in its negative sense) is important for studying valency of words (Hajičová and Panevová 1984). The following example searches on the tectogrammatical layer of PDT. We want a Predicate that governs directly an Actor and a Patient and nothing else (directly):

```
[functor=PRED,_#sons=2]([functor=ACT],[functor=PAT]).
```

The graphical representation of the query is:



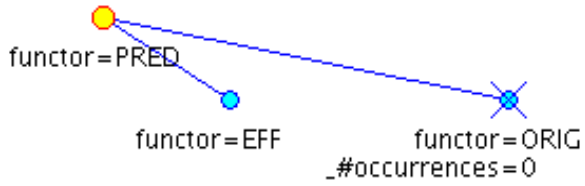
If we replaced PAT with ADDR, we might search for errors in the evaluation, since the theory

forbids Actor and Addressee being the only parts of a valency frame.

So far, we could only restrict number of nodes. But we often want to restrict a presence of a certain type of node. We want to specify that there is not a node of a certain quality. For example, we might want to search (again on the tectogrammatical layer) for an Effect without an Origo in a valency frame. The meta-attribute that allows this real type of negation is called `_#occurrences`. It controls the exact number of occurrences of a certain type of node, in our example of Origos:

```
[functor=PRED]([functor=EFF],
[functor=ORIG,_#occurrences=0])
```

with graphical representation:



It says that the Predicate has at least one son – an Effect, and that the Predicate does not have an Origo son.

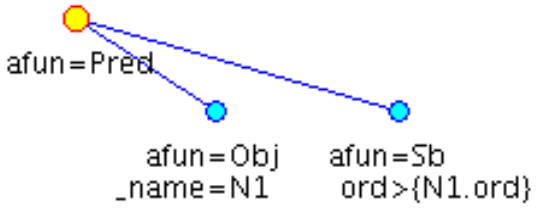
There is still one important thing that we cannot achieve with the meta-attributes presented so far. We cannot set any relation (other than dependency) between nodes in the result trees (such as order, agreement in case, coreference). All this can be done using the meta-attribute `_name` and a system of references. The meta-attribute `_name` simply names a node for a later reference from other nodes.

Curly brackets enclose a reference to a value of an attribute of the other node (with a given name) in the result tree. This, along with the dot-referencing inside the reference and some arithmetic possibilities, completes our description of the syntax of the query language from subsection 2.1.

In the following example (back on the analytical layer and knowing that attribute `ord` keeps the order of the nodes (~ tokens) in the tree (~ sentence)) from left to right, we search for a Subject that is on the right side from an Object (in the tree and also in the sentence):

```
[afun=Pred]
([afun=Sb,ord>{N1.ord}],
[afun=Obj,_name=N1])
```

with graphical representation:



We have named the Object node `N1` and specified that ord of the Subject node should be bigger than `ord` of the `N1` node. If we used `ord>{N1.ord}+5`, we would require them to be at least five words apart.

Meta-attribute _#lbrothers (_#rbrothers) contains number of left (right) brothers of a particular node in the result tree. Thus, we can define that a node (e.g. an Attribute) is the leftmost son of another node (e.g. an Object):

```
[afun=Obj]([afun=Atr,_#lbroth-
ers=0]).
```

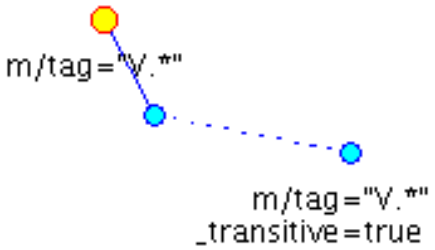
Meta-attribute `_transitive` defines a transitive edge. The following example searches for a verb node that governs transitively another verb node:

```
[m/tag="V.*"]([m/tag="V.*",_tran-
sitive=true]).
```

If we do not want them to be direct father and son, we have two possibilities: Either we put another node without any evaluation in between them in the query:

```
[m/tag="V.*"]([]([m/tag="V.*",
_transitive=true]))
```

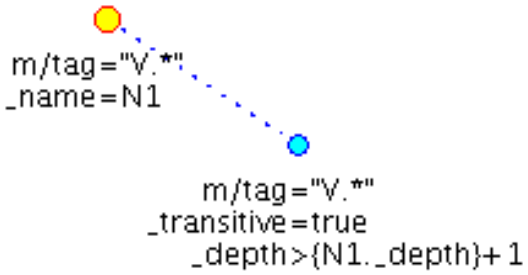
with graphical representation:



or, we can use meta-attribute `_depth` and references:

```
[m/tag="V.*",_name=N1]
([m/tag="V.*",_transitive=true,
_depth>{N1._depth}+1])
```

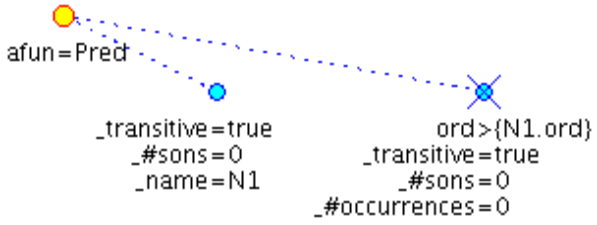
which is perhaps a little bit more complex. The graphical representation of the query is:



Using several meta-attributes in one query can form a powerful combination. The following example searches for the rightmost list descendant of a Predicate:

```
[afun=Pred]([_transitive=true,
_#sons=0,_name=N1],[_transi-
tive=true,_#sons=0,
ord>{N1.ord},_#occurrences=0])
```

with graphical representation:



The first transitive descendant of the Predicate is the list (`_#sons=0`) we are looking for. The second transitive descendant is a list that we do not want to be in the result (with higher `ord`). Therefore, we set `_#occurrences` to zero.

3 Conclusion

We have presented Netgraph query language, its basics and also its advanced techniques, namely meta-attributes, references and their combination.

We have demonstrated that many linguistic phenomena can be searched for using this language. It can be shown (Mírovský 2008) that Netgraph querying power outperforms the querying power of TGrep (Pito 1994), which is a traditional (and nowadays outdated) treebank searching tool. On the other hand, it seems (it has not been studied thoroughly yet) that Netgraph has slightly lesser searching power than TGrep2 (Rohde 2005), which can use any boolean combination of its searching patterns.

Acknowledgement The research reported in this paper was supported by the Grant Agency of the Academy of Sciences of the Czech Republic, project IS-REST (No. 1ET101120413).

References

Hajič J. et al. 2006. Prague Dependency Treebank 2.0. *CD-ROM LDC2006T01, LDC, Philadelphia, 2006.*

Hajič J. 2004. Complex Corpus Annotation: The Prague Dependency Treebank. *Jazykovedný ústav Ľ. Štúra, SAV, Bratislava, 2004.*

Hajič J., Vidová-Hladká B., Panevová J., Hajičová E., Sgall P., Pajas P. 2001a. Prague Dependency Treebank 1.0 (Final Production Label). *CD-ROM LDC2001T10, LDC, Philadelphia, 2001.*

Hajič J., Pajas P. and Vidová-Hladká B. 2001b. The Prague Dependency Treebank: Annotation Structure and Support. *In IRCS Workshop on Linguistic databases, 2001, pp. 105-114.*

Hana J., Zeman D., Hajič J., Hanová H., Hladká B., Jeřábek E. 2005. Manual for Morphological Annotation, Revision for PDT 2.0. *ÚFAL Technical Report TR-2005-27, Charles University in Prague, 2005.*

Hajič J. et al. 1997. A Manual for Analytic Layer Tagging of the Prague Dependency Treebank. *ÚFAL Technical Report TR-1997-03, Charles University in Prague, 1997.*

Hajič J. 1998. Building a Syntactically Annotated Corpus: The Prague Dependency Treebank. *In Issues of Valency and Meaning, Karolinum, Praha 1998, pp. 106-132.*

Hajičová E., Havelka J., Sgall P., Veselá K., Zeman D. 2004. Issues of Projectivity in the Prague Dependency Treebank. *MFF UK, Prague, 81, 2004.*

Havelka J. 2007. Beyond Projectivity: Multilingual Evaluation of Constraints and Measures on Non-Projective Structures. *In Proceedings of ACL 2007, Prague, pp. 608-615.*

Hajičová E. 1998. Prague Dependency Treebank: From analytic to tectogrammatical annotations. *In: Proceedings of 2nd TST, Brno, Springer-Verlag Berlin Heidelberg New York, 1998, pp. 45-50.*

Hajičová E, Panevová J. 1984. Valency (case) frames. *In P. Sgall (ed.): Contributions to Functional Syntax, Semantics and Language Comprehension, Prague, Academia, 1984, pp. 147-188.*

Mírovský J. 2006. Netgraph: a Tool for Searching in Prague Dependency Treebank 2.0. *In Proceedings of TLT 2006, Prague, pp. 211-222.*

Hajičová E., Partee B., Sgall P. 1998. Topic-Focus Articulation, Tripartite Structures and Semantic Content. *Dordrecht, Amsterdam, Kluwer Academic Publishers, 1998.*

Kučová L., Kolářová-Řezníčková V., Žabokrtský Z., Pajas P., Čulo O. 2003. Anotování koreference v Pražském závislostním korpusu. *ÚFAL Technical Report TR-2003-19, Charles University in Prague, 2003.*

Ondruška R. 1998. Tools for Searching in Syntactically Annotated Corpora. *Master Thesis, Charles University in Prague, 1998.*

Mírovský J., Ondruška R., Průša D. 2002. Searching through Prague Dependency Treebank - Conception and Architecture. *In Proceedings of The First Workshop on Treebanks and Linguistic Theories, Sozopol, 2002, pp. 114--122.*

Mírovský J.: Netgraph Home Page: *http://quest.ms.mff.cuni.cz/netgraph*

Mírovský J. 2008. Towards a Simple and Full-Featured Treebank Query Language. *In Proceedings of First International Conference on Global Interoperability for Language Resources, Hong Kong, 2008, in print.*

Pito R. 1994. TGrep Manual Page. *Available from http://www.ldc.upenn.edu/ldc/online/treebank/*

Rohde D. 2005. TGrep2 User Manual. *Available from http://www-cgi.cs.cmu.edu/~dr/TGrep2/tgrep2.pdf*