Learning to Search in Prague Dependency Treebank

Jiří Mírovský and Jarmila Panevová
Charles University in Prague
Institute of Formal and Applied Linguistics
{mirovsky|panevova}@ufal.mff.cuni.cz

Abstract

We present Netgraph – an easy to use tool for searching in linguistically annotated treebanks. On several examples from the Prague Dependency Treebank we introduce the features of the searching language and show how to search for some frequent linguistic phenomena.

1. Introduction

Searching in a linguistically annotated treebank helps linguistic research not only in the field of computational linguistics but also in the theoretical linguistics. There exist various tools for searching in treebanks and they vary in the amount of mathematical knowledge or programming skills they require from users.

Netgraph is a searching tool designed to be as simple as possible. It tries to be easy to understand, learn and use. The query language is very intuitive and motivated by linguistic needs. It follows the idea "what you see is what you get", or rather "what you want to see in the result is what you draw in the query". Of course, to be able to search in a treebank using Netgraph, the user has to know the system of annotation of the treebank.

Netgraph has been primarily developed for searching in two structured layers of the Prague Dependency Treebank (PDT) – the analytical layer (which is close to the surface syntax) and the tectogrammatical layer (the deep syntax). Searching is possible on both the layers separately and also in combination across the layers boundary. In this paper, we will use examples from both the analytical and the tectogrammatical layers.

In *section 2* we very briefly describe the Prague Dependency Treebank, just to make the examples in the subsequent text more understandable. Anyone familiar with this treebank may safely skip this section.

In *section 3* we mention in a few words the history of Netgraph and its properties as a tool.

In *section 4* we introduce 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 5* we offer some concluding remarks.

2. The Prague Dependency Treebank

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). 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. Non-projective constructions (that are quite frequent both in Czech (Hajičová et al. 2004) and in some other languages (Havelka 2007)) are allowed. Analytical functions are kept at nodes (attribute `a/afun`), but in fact they are names of the dependency relations between a dependant (son) node and its governor (father) node.

The tectogrammatical layer captures the linguistic meaning of the sentence in its context. Again, the annotation is a dependency tree with labeled nodes. The correspondance 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.

Many nodes found on the analytical layer disappear on the tectogrammatical layer (such as functional words, e.g. prepositions, subordinating conjunctions, etc.). The information carried by these nodes is stored in attributes of the remaining (autosemantic) nodes and can be reconstructed. On the other hand, some nodes representing for example obligatory positions of verb frames, deleted on the surface, and some other deletions, are regenerated on this layer (for a full list of deletions, see Mikulová et al. 2006).

The tectogrammatical layer goes beyond the surface structure and corresponds to the semantic structure of the sentence, replacing notions such as Subject and Object by functors like Actor, Patient, Addressee etc (see Hajičová 1998, for a full list of functors, see Mikulová et al. 2006).

Attribute `functor` describes the dependency between a dependant node and its governor and again is stored at the son-nodes. A tectogrammatical lemma (attribute `t_lemma`) is assigned to every node. Grammatemes are rendered as a set of 16 attributes grouped by the "prefix" `gram` (e.g. `gram/verbmod` for verbal modality).

The total of 39 attributes are assigned to every non-root node of the tectogrammatical tree, although (based on the node type) only a certain subset of the attributes is necessarily filled in.

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`). It is in general different from the surface word order, and all the resulting trees are projective by the definition of deep word order.

To be complete (as much as possible in this short description), let us add that 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 `id`s of coreferential nodes of the respective types.

3. Netgraph As a Tool

The development of Netgraph started in 1998 as a topic of Roman 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, taking into account a new system of annotation and feedback from the users. Now it is a fully functional tool for complex searching in PDT 2.0.

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 programming language) 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. It 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 desctiption of the inner architecture of Netgraph and of the communication between the server and the client was given in Mírovský, Ondruška and Průša (2002).

4. The Query Language

In this chapter, we describe the query language of Netgraph. We start from very simple examples and proceed to more complex ones.

4.1. 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, the tree becomes part of the result. The query has both a textual form and a graphical form. For lack of space, we will use its textual form in this paper. Each textual query has its full graphical counterpart.

The simpliest possible query (and of little interest) 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].*"]*[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 both a Subject and at least one Object: *[afun=Pred]([afun=Sb],[afun=Obj])*[2]. 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 parent. It does not prevent other nodes to be directly governed by the Predicate either.

4.2. Meta-attributes

This simple query language, described briefly in only a few examples, is quite useful but not powerful enough. There is no real possibility of setting more complex negation, no way of restricting the position of the query tree 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 data but they pretend to be ordinary attributes and in fact are treated the same way like normal attributes. There are about ten of them, each adding some power to the query language, enhancing its semantics, while keeping the syntax of the language on the same simple level. We will present several of the meta-attributes, along with the linguistic motivation that lead to their addition.

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 coordinating node in the queries, we have introduced the meta-attribute *_optional* that marks an optional node. It then may but does not have to appear 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=true]([afun=Obj]))*. The Coordination node

1 We do not expect to find many real counterexamples. In any case, square brackets enclose a node, attributes are separated by a comma, quotation marks enclose a regular expression.

2 Parentheses enclose a subtree of a node, brothers are separated by a comma

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 node.

There is a group of meta-attributes of rather technical nature. They allow setting a position of the query tree 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 et al. 1984). The following example searches on the tectogrammatical layer of PDT. There is a Predicate that governs directly an Actor and a Patient *and nothing else (directly): [functor=PRED,_#sons=2]([functor=ACT],[functor=PAT])*. 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.

4.3. Negation

So far, we only could restrict number of nodes. But we often want to restrict a presence of a certain *type* of node. We want to specify that a node of a certain quality is not present at a particular place in the result tree. For example, we might want to search (again on the tectogrammatical layer) for a Predicate governing an Effect but not an Origo. 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])*. It says that the Predicate has at least one son – an Effect, and that the Predicate does not have an Origo son.

4.4. References to Other Nodes

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 (for word-order studies), 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. In the following example (back on the analytical layer and knowing that attribute *ord* keeps the order of the node (~ token) in the tree (~ sentence)), we search for a Subject that is on the right side from an Object:
[afun=Pred]([afun=Sb,ord>{N1.ord}],[afun=Obj,_name=N1])[3]. We have named the Object node

3 Curly brackets enclose a reference to a value of an attribute of another node (with a given name) in the result tree.

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.

4.5. Linear Searching

Sometimes we only know what sequence of words we are searching for and do not know how or do not want to bother to write a structured query. For this, there is a meta-attribute *_sentence*. If we were to search for all trees containing the phrase „*v souvislosti s*“, we might simply write a query: *[_sentence=".*[Vv] souvislosti s.*"]*. Nevertheless, it serves only as a simplification of these special cases, Netgraph remains a tool for searching in tree structures.

5. Conclusion

This has only been a very quick glance at the Netgraph query language. Nevertheless, we hope that we have been successful in showing that it allows to write quite complex queries in a simple way. Although it has been primarily designed for searching in Prague Dependency Treebank, the query language is general and it can be used for other treebanks and other languages, too.

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), and by the Grant Agency of the Czech Republic, project 405/06/0589.

References

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

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.*

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. et al. (2006): Prague Dependency Treebank 2.0. *CD-ROM LDC2006T01, LDC, Philadelphia, 2006*.

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.*

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

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.*

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.*

Mikulová M., Bémová A., Hajič J., Hajičová E., Havelka J., Kolářová V., Kučová L., Lopatková M., Pajas P., Panevová J., Razímová M., Sgall P., Štěpánek J., Urešová Z., Veselá K., Žabokrtský Z. (2006): Annotation on the tectogrammatical level in the Prague Dependency Treebank. Annotation manual. *Tech. Report 30, ÚFAL MFF UK, 2006.*

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

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.*

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