Automatic Functor Assignment in the Prague Dependency Treebank *

Zdeněk Žabokrtský

Czech Technical University, Department of Computer Science
121 35 Praha 2, Karlovo nám. 13, Czech Republic
zabokrtz@cs.felk.cvut.cz

Abstract. This paper presents work in progress, the goal of which is to develop a module for automatic transition from analytic tree structures to tectogrammatical tree structures within the Prague Dependency Treebank project. Several rule-based and dictionary-based methods were combined in order to be able to make maximal use of both information extractable from the training set and a priori knowledge. The implementation of this approach was verified on a testing set, and a detailed evaluation of the results achieved so far is presented.

1 Introduction

The process of syntactic tagging in the Prague Dependency Treebank (PDT) is divided into two steps: The first step results in *analytic tree structures* (ATS), in which every word form and punctuation mark is explicitly represented as a node of rooted tree, with no additional nodes added (except for the root of the tree of every sentence). The second step results in *tectogrammatical tree structures* (TGTS), which approximate the underlying sentence representations according to [4]. In contrast to the ATSs, only autosemantic words have nodes of their own in TGTSs, informations about functional words (prepositions, subordinating conjunctions etc.) are contained in the tags attached to the autosemantics nodes. Figure 1 depicts an example of a TGTS.

Besides slight changes in the topology of the input ATS (for instance, pruning of synsemantic nodes), the transition from ATSs to TGTSs involves the assignment of the tectogramatical function (*functor*) to every node in the tree. There are roughly 60 functors divided into two subroups (cf. [4]): (i) *actants* (ACTor, PATient, ADDRessee, EFFect, ORIGin) and (ii) *free modifiers*: TWHEN (time-when), LOCaction, MEANS, EXTent, BENeficiary, ATTribute ...).

Presently, the topological conversion and the assignment of a few functors (e.g., ACT, PAR, PRED) are solved automatically by the procedure of Böhmová et al. [1]. However, most of the functors have to be assigned manually. The

* I would like to thank my advisor Ivana Kruijff-Korbayová for her permanent support. I am also indebted to Alevtina Bémová for consultations about functor assignment, to Petr Pajas for the help with the data preprocessing and to Julia Hockenmaier for her useful comments.

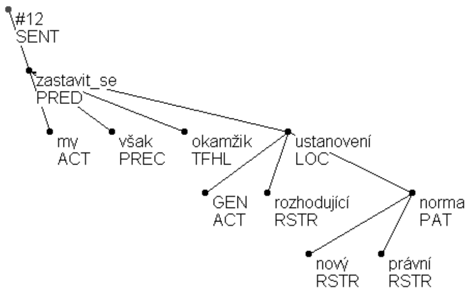

Fig. 1. TGTS of the sentence *Zastavme se však na okamžik u rozhodujících ustanovení nové právní normy.* (Let's however stop for a moment at the most important paragraphs of the new legal norm.)

amount of labor involved in the manual annotation obviously slows down the growth of the PDT on the tectogrammatical level. Decreasing the amount of manual annotation has been the motivation for developing the more complex *automatic functor assignment system* (AFA) presented in this paper. Let us describe the starting position.

– No general unambiguous rules for functor assignment are known, human annotators use mostly only their language experience and intuition. We cannot reach 100% correctness of AFA since even the results of individual annotators sometimes differ.
– The annotators usually use the whole sentence context for their decision. It has not been measured how often it is really unavoidable to take the full context into account or how large the context must be.
– Preliminary measurements revealed that the distribution of functors is very non-uniform. The 15 most frequent functors cover roughly 90% of nodes. Conversely, there are hardly any examples for the least frequent functors.
– It would be very time consuming to test the performance AFA on randomly selected ATSs and find errors manually. Fortunately we can use the ATSs for which manually created TGTSs are already avaliable, annotate them automatically and compare the results against the manually annotated TGTSs.
– The available TGTSs contain imperfect data. Some errors are inherited from ATSs, and functor assignments are in some cases ambiguous (nodes with more than one functor) or incomplete (some nodes have no functor yet).

2 Materials

Training and Testing Sets When I started working on AFA, 18 TGTS files were available, each containing up to 50 sentences from newspaper articles. This was a sufficient amount of data for mining knowledge, which can improve the AFA's performance. But in order to reliably measure AFA's correctness, it is necessary to have a separate data set not used for knowledge mining. Therefore I randomly selected 15 files for the training set and 3 files for the testing set. After removing incomplete and ambiguously assigned nodes, the training set contained 6049 annotated nodes, and the testing set 1089 annotated nodes.

Data Preprocessing Neither the maximum degree of a node (i.e., the number of outgoing edges) nor the depth of a tree are limited in TGTSs. The trees thus can be very complex, and working with the whole tree context of the individual nodes would make AFA unnecessarily complicated. For the sake of the experiments described here, I assumed that reasonable correctness can be achieved using only information about the node to be annotated and about its governing node (i.e., about the edge in the tree). So the first step of the preprocessing was the *transformation from the tree structure into the list of edges.*

Each node in PDT can have tens of attributes, majority of them beeing useless for AFA. Hence, a selection of the relevant attributes is performed next (*feature selection*). I chose the following set: word form, lemma, full morphological tag and analytical function of both the governing and dependent node, preposition or conjunction which binds the governing and the dependent node, and the functor of the dependent node.

In order to make the subsequent processing easier, 3 additional simple attributes (the parts of speech of both nodes, the morphological case of the dep. node) were extracted from these 10 attributes (*feature extraction*). Finally, each accented character has been substituted with the corresponding ASCII character followed by "_". Having a vector of 13 symbolic attributes, the task of AFA can be now formulated as the *classification of the symbolic vectors into 60 classes.*

3 Implementation

The AFA system has been designed as a collection of small programs written mostly in Perl. Each method of functor assignment forms a separate program (script), the data to be assigned goes through a sequence of these scripts in a pipeline fashion. Each method can assign only those nodes which have not been assigned by any of the previous scripts yet. This approach enables flexible tuning of the AFA characteristics (precision, recall) simply by reordering or removing the individual methods. This advantage would be lost in the case of one compact complicated program.

Rule-based Methods (RBM) The RBMs consist of simple hand written decision trees. They use no external data and therefore are independent of the quality of the training set. They do not bring any new information into the PDT, only transform the information contained in an ATS. Currently I have 7 methods with reasonable precision:

1. `verbs_active`: if the governing node is a verb in active form then
   - if the analytical function (afun) is subject, then the node is assigned the functor ACT ($\rightarrow$ ACT)

   - if afun is object and case is dative then → ADDR
   - if afun is object and case is accusative then → PAT
2. `verbs_passive`: if the governing node is a verb in passive form:
   - if afun is subject then → PAT
   - if afun is object and case is dative then → ADDR
   - if afun is object and case is instrumental then → ACT
3. `adjectives`: if the node corresponds to an adjective
   - if it is a possessive adjective then → RSTR
   - else → RSTR
4. `pronounposs`: if the node is a possessive pronoun then → APP
5. `numerals`: if the node is a numeral then → RSTR
6. `pnom`: if afun is PNOM then → PAT
7. `pred`: if afun is PRED then → PRED

Dictionary-based Methods (DBM) It is not feasible to resolve all the remaining unassigned functors using only simple RBMs like those above, since we could not profit from the growing volume and diversity of the training set.

I have so far developed four methods using different types of dictionaries:

- `adverbs`: The couples *adverb–functor* were automatically extracted from the training set, and added to the list of adverbs from [2]; from the combined list, the *unambiguous* (accompanied always with the same functor) adverbs were extracted. Such a dictionary can be used to assign functors to adverbs. Examples from the dictionary: *výlučně* (exclusively) RHEM, *výrazně* (extensively) EXT
- `subconj`: A dictionary of unambiguous *subordinative conjunctions* was constructed in the same way as the dictionary of adverbs. If a verb is related to its governing node by one of these conjunctions, the functor can be easily assigned.
  Examples from the dictionary: *i když* (even when) CNCS,
  jelikož (because) CAUS, *jen co* (as soon as) TWHEN, *jestli* (if) COND
- `prepnoun`: All the *preposition–noun* pairs (a preposition followed by a noun) were extracted from the training set. The unambiguous couples which occured at least twice were inserted into the dictionary.
  Examples from the dictionary: *v roce* (in year) TWHEN, *pro podnikatele* (for businessman) BEN, *od doby* (from time) TSIN, *z odvětví* (from branch) DIR1, *v zemích* (in countries) LOC
- `similarity`: The dictionary is formed by the entire training set. The functor of the most similar vector found in the training set is used for assignment. The (in)equality of individual attributes has different impact (weight) on the similarity function, e.g., the part of speech is more important than the lemma. The weights were determined experimentally. Example: for *zálohy na daně* (pre-payments of taxes), where the dependent node *daně* (taxes) is to be assigned a functor, the most similar record found is *návrh na stanovení* (proposal of determination), so the functor PAT of the dependent node is used.

4 Results

Testing Set The testing set was not used in the mining of knowledge (dictionaries), therefore we can apply both rule-base and dictionary method on it. For each method, six quantitative characteristics have been determined (Table 1):

- *Cover* = the number of all nodes assigned by the given method
- *Relative cover* = cover divided by number of all functors to be assigned (1089 in the training set). This number also reflects the frequency of several phenomenona (e.g., possessive pronouns).
- *Errors* = the number of incorrectly assigned functors
- *Hits* = the number of correctly assigned functors
- *Recall* = the percentage of correct functor assignments by the given method among all functors to be assigned (hit/1089·100%)
- *Precision* = the percentage of of correct functor assignments by the given method among all functors assigned by this method (hits/cover·100%)

Method	Cover	Rel. cover	Hits	Recall	Errors	Precision
pred	104	9.5 %	104	9.6 %	0	100 %
verbs_active	199	18.3 %	184	16.9 %	15	92.5 %
verbs_passive	7	0.6 %	6	0.6 %	1	85.7 %
pnom	34	3.1 %	32	2.9 %	2	94.1 %
adjectives	177	16.3 %	170	15.6 %	7	96.0 %
numerals	21	1.9 %	15	1.4 %	6	71.4 %
pronounpos	16	1.5 %	13	1.2 %	3	81.3 %
subconj	3	0.3 %	2	0.2 %	1	66.7 %
adverbs	34	3.1 %	30	2.8 %	4	88.2 %
prepnoun	9	0.8 %	9	0.8 %	0	100 %
similarity	485	44.5 %	287	26.4 %	198	59.2 %
Total	Σ=1089	Σ=100 %	Σ=852	Σ=78.2 %	Σ=237	78.2 %

Table 1. Results of AFA on the testing set

The methods in Table 1 are sorted in the same order as they were executed. This order permutation reaches the highest precision. The `similarity` method is handicapped by the fact that all easily solvable cases are assigned by its predecessors. If we use `similarity` alone, we get Recall=Precision=73%.

I believe that the results of this first implementation are satisfactory, I had expected lower overall precision. However, I cannot compare it to anything else, since there is no other AFA implementation with comparable recall within the PDT project.

Rule-based Methods on Training Set In order to verify the precision of RBMs, we can apply them on the training set as well (see Table 2). Note that the size of the training set is 6049 nodes.

Method	Cover	Rel. cover	Hits	Recall	Errors	Precision
pred	574	9.5 %	554	9.2 %	20	96.5 %
verbs_active	973	16.1 %	907	15.0 %	66	93.2 %
verbs_passive	34	0.6 %	27	0.4 %	7	79.4 %
pnom	164	2.7 %	152	2.5 %	12	92.7 %
adjectives	1063	17.6 %	976	16.1 %	87	91.8 %
numerals	92	1.5 %	66	1.1 %	26	71.7 %
pronounpos	64	1.1 %	61	1.0 %	3	95.3 %
Total	Σ=2964	Σ=49.0 %	Σ=2743	Σ=45.3 %	Σ=221	92.5 %

Table 2. Results of RBMs on the training set

Precision versus Recall We have to decide whether we prefer to *minimize the number or errors* (maximizing precision using only the methods with the best precision) or *maximize the number of correctly assigned nodes* (maximizing recall using all the methods with admissible precision). The optimal compromise should be influenced by the *misclassification cost* which can be estimated as an amount of annotators' work for finding and correcting incorrectly assigned functors.

5 Conclusion and Future Work

I implemented several methods for automatic functor assignment, tested them and evaluated their characteristics. Methods based on rules had higher precision than dictionary-based methods. The possibility of combining individual approaches opened the question whether we prefer to assign, e.g., 49 % of functors with 92 % precision or to assign everything with 78 % precision.

All the available TGTSs are from newspaper articles. The distance between the training set and the testing set is thus rather small. If AFA were to be applied to other than newspaper articles, it is likely that precision would be slightly lower.

As more manually annotated TGTSs become available, we can expect improvements of the dictionary-based methods. Moreover, it will hopefully be possible to discover some more rules for functor assignment using the machine learning system C4.5; we have so far obtained promising preliminary results.

References

1. Böhmová, A., Panevová, J., Sgall, P.: *Syntactic Tagging: Procedure for the Transition from the Analytic to the Tectogrammatical Tree Structures.* Text, Speech and Dialogue, Springer (1999)
2. Hajičová, E., Panevová, J., Sgall, P.: *Manuál pro tektogramatické značkování.* ÚFAL MFF UK (1999)
3. Panevová, J.: *Formy a funkce ve stavbě české věty.* Academia (1980)
4. Petr Sgall, Eva Hajičová, and Jarmila Panevová: *The Meaning of the Sentence in its Semantic and Pragmatic Aspects.* Reidel, Dordrecht, The Netherlands, 1986.