A Machine Learning Approach to Automatic Functor Assignment in the Prague Dependency Treebank

Zdeněk Žabokrtský[*] **Petr Sgall**[*] **Sašo Džeroski**[†]

[*] Center for Computational Linguistics
Charles University, Prague, Czech Republic
`{zabokrtsky,sgall}.ufal.mff.cuni.cz`
[†]Jožef Stefan Institute
Ljubljana, Slovenia
`saso.dzeroski@ijs.si`

Abstract

The aim of this paper is to describe and evaluate a system that automates a part of the transition from analytical to tectogrammatical tree structures within the Prague Dependency Treebank. In particular, it assigns functors to autosemantic words. The system is based on the machine learning approach of decision tree induction. The resulting software tool is incorporated into the annotation process and significantly reduces the manual annotation effort during the transition from analytical tree structures to the tectogrammatical tree structures, which consumes a huge amount of time of linguistic experts.

1. Introduction

1.1. Prague Dependency Treebank

The Prague Dependency Treebank is a research project at the Center for Computational Linguistics[1], Charles University, Prague. It is aimed at a complex annotation of a part of the Czech National Corpus (CNC). The sentences from the CNC are assigned their underlying representations in three steps of annotation: morphological, analytical, and tectogrammatical. PDT thus differs from most annotation schemes in that it reaches representations suitable as input for a semantic interpretation, making use of a longterm theoretical research taking into account most different relationships between (dependency-based) syntax and (intensional) semantics, see (Hajičová et al.,1998), (Hajičová et al.,2002).

At the morphological level, a tag and a lemma are assigned to each word form from the input text. At the analytical level, the linear sequence of words and punctuation marks is enriched with a dependency structure representing the given sentence. In the resulting analytical tree structure (ATS), each node is assigned its analytical function (such as Subject, Object, Adverbial, Attribute).

During the transition from ATSs to tectogrammatical tree structures (TGTSs), the topology of the tree is changed. The difference can be seen in Figure 1 (a) and (b) (see Figure 2 for one more example of TGTS). Neither function words (prepositions, auxiliaries, conjunctions except coordinating conjunctions), nor punctuation marks have their own nodes in a TGTS; their counterparts are captured in the attributes of the remaining nodes, representing the lexical words. Besides lexical words, there are also nodes for coordination and apposition.

Each of these words is annotated with values of its grammatemes, i.e. morphological attributes as number, tense, modality, and its functor, which represents its syntactic position within the sentence, e.g., Actor, Patient, Addressee, Effect, Origin, various types of spatial and tempo-ral circumstantials, Means, Manner, Condition, etc. There are approximately 60 functors (arguments or theta roles, and adjuncts). Functors provide a detailed information on the relation between the given node and its governing node. See the appendix for the list of the most frequent functors and their examples.

The examples of TGTSs are given in Figures 1(b) and 2. The respective sentences are as follows:

(1)   Michálková upozornila, že zatím   je
lit.: Michálková pointed-out that meanwhile is
zbytečné   podávat na správu   žádosti
superfluous to-submit to administration requests
či žádat ji o podrobnější   informace.
or to-ask it for more-detailed information.

'M. pointed out that for the time being it was superfluous to submit requests to the administration, or to ask it for a more detailed information.'

(2)   Hodnotná a   okouzlující kulturní událost je
lit.: Valuable and fascinating cultural event   is
důkazem, že dlouhodobě kvalitní
proof   that longterm   high-quality
dramaturgie výstav   Malovaného domu,
dramaturgy of-exhibitions of-Painted House,
kterou   vytváří Luboš   Kressa,   na
which.accus creates Luboš.nom Kressa.nom to
sebe váže i   další umělecké a   kulturní
itself binds also further artistic   and cultural
aktivity.
activities.

'The valuable and fascinating cultural event documents that the long-term high-quality strategy of the Painted House exhibitions, established by L. K., attracts further activities in the domains of art and culture.'
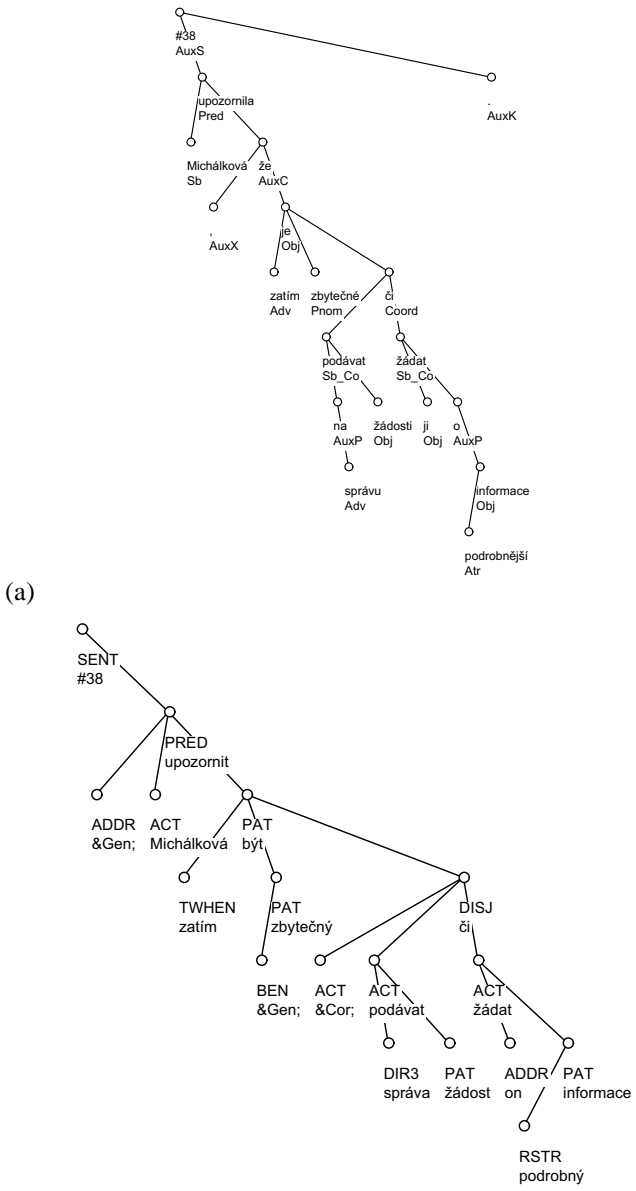
---

[1]`http://ckl.mff.cuni.cz`

(a)

(b)

Figure 1: The analytical tree structure (a) of sentence (1) and the associated tectogrammatical tree structure (b). In the analytical tree, the nodes are labeled with their word forms and analytical functions, in the tectogrammatical tree the nodes are labeled with their lemmas and functors. Note that nodes are added (marked by the sign &) for (i) General Addressee with *upozornit* 'point out sth. to so.' and General Benefactive with *zbytečný* 'superfluous for so.', and (ii) the zero Actor of the infinitives *podávat* 'to submit' and *žádat* 'to ask for', which is controlled by (i.e. referentially identical with) the Benefactive of *být zbytečný* 'to be superfluous'. The position of the node with the label *či* 'or' illustrates the handling of coordinated constructions.

1.2. Related Work

The first realization of AFAS (Automatic Functor Assignment System, (Žabokrtský, 2000)) was based on a combination of different families of "assigners", in particular of handcrafted rules and dictionary-based methods. The system was later (Džeroski, Žabokrtský, 2001) enriched with
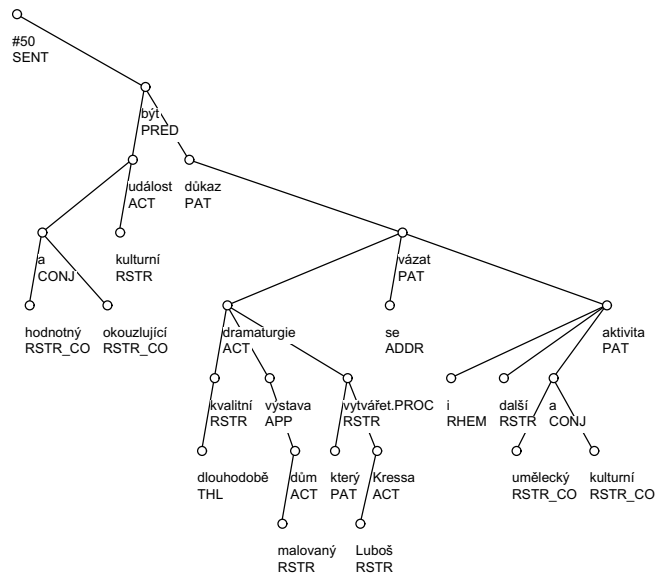

Figure 2: Example of the tectogrammatical tree structure. The nodes are labeled with their lemmas and functors. Note the position of a focus sensitive operator (rhematizer) *i* 'also'.

a method based on Machine Learning (ML), namely on the decision trees generated by C4.5 (Quinlan, 1993). The AFAS described in this paper differs from the previous attempts in the fact that it is *exclusively based on ML*, thus forming a monolithical, easy-to-retrain system.

2. Learning from the Tectogrammatical Tree Data

2.1. Reformulation of the Task of AFAS from the ML Perspective

The task for an automatic functor assignment system (AFAS) can be formulated as follows: given (i) the topology of a TGTS and (ii) the values of the morphological and analytical attributes of each node, assign a functor to each node.

Since treating the tree as a whole would make the learning task very difficult, we approximate it with the following reformulation. For each node N of the tree, the task is to assign a functor to node N (a coordination node may appear between M and N, but it does not influence the functor of N), taking into account only the values of its attributes and the values of attributes of the nearest autosemantic node M that governs node N. In other words, we are to classify the edges of the tree (or short vertical paths in the tree in case there is an intermediate coordination node, like between words *kulturní* and *aktivita* in Figure 2). Each class corresponds to one functor.

2.2. Available Material

When the experiments described in this paper were performed, the manual annotation of 43 files had been finished on the tectogrammatical level. Each file contains up to 50 sentences, there were altogether 1536 sentences in

these files. The total number of autosemantic and coordination nodes in the sentences were 28216. We excluded the nodes with unambiguously assigned functors, and the nodes which were manually added. The added nodes have no surface counterpart word, although they must be present in the underlying structure (for instance, ellipses of actor etc.). 27463 nodes remained for the training and testing purposes.

2.3. Data Preprocessing

2.3.1. Vectorizing the Tree

All the information about a node is contained in its attributes from the morphological level (lemma, morphological tag), analytical level (analytical function), and tectogrammatical level (functor). As we said above, for the decision about the functor of a node, we take into account only the information about the node itself and the information about its nearest governing autosemantic node. Therefore we approximate the functor assignment with a mapping the domain of which is formed by a Cartesian product of values of the morphological and analytical attributes of both nodes, and the range of which is equal to the set of all functors. Since the number of node attributes is limited, we can put the attributes into a vector. Currently we create for each node the vectors consisting of the following attributes:

1. *lexical attributes* - lemmas of both nodes, and possibly also the lemma of a preposition or a subordinating conjunction that binds both nodes,

2. *morphological attributes* - for both nodes, we extract the following attributes from their morphological tags: part of speech, so called subpart of speech, morphological voice, morphologic case,

3. *analytical attributes* - the analytical functions of both nodes,

4. *topological attributes* - number of children (directly depending nodes) of both nodes in the TGTS,

5. *ontological attributes* - semantic position of the node lemma within the EuroWordNet Top Ontology.

Wherever we use attribute names in the following text, the names of the attributes extracted from the governing node start with prefix g_, while the names of the attributes extracted from the dependent node starts with d_.

2.3.2. Ontological Attributes from EuroWordNet

The Top Ontology used in EuroWordNet (EWN) contains the (structured) set of 63 basic semantic concepts like Place, Time, Human, Group, Living, etc. For the majority of English synsets (set of synonyms, the basic unit of EWN), the appropriate subset of these concepts are listed. Using the Inter Lingual Index that links the synsets of different languages, the set of relevant concepts can be found also for Czech lemmas. The set of ontological attributes then comprises 63 binary attributes determining the positive or negative relation between the (Czech) lemma and the semantic concepts.

2.4. C5 Learning

2.4.1. Learning Decision Trees

Decision tree induction is one of the most popular machine learning approaches. It takes as input a set of examples (represented as vectors of feature values and their classifications) and produces a tree-like structure called *decision tree* that can be used for classifying new examples. Internal nodes in the tree correspond to features (also called attributes), branches correspond to feature values, and leaves of the tree correspond to classifications (predicting specific class values).

C5.0 is a successor of C4.5 (Quinlan, 1993), which is probably the most widely used program for inducing decision trees. It implements the TDIDT (Top Down Induction of Decision Trees) approach, where a feature is first selected that discriminates best among the class values of the given training examples. Once this feature is selected, it is assigned to the root of the tree; the examples are partitioned according to the values of this feature and tree construction is repeated recursively; the resulting subtrees are attached to the branches of the root node. If the set of examples contains examples of only one class or if no good attribute can be found to split on, a leaf is created which predicts a specific class value. This is the criterion for terminating the recursion. C5.0 also performs pruning (simplification) of the induced decision trees. The subtrees that are built on a too small number of examples to be statistically reliable are removed and replaced by leaves (in other words, they are pruned). Several parameters control the construction of a decision tree with C5.0 (e.g., by setting the degree of pruning): we used the default setting of their values.

2.4.2. Rulesets

An important feature of C5.0 is its mechanism to convert trees into collections of rules called *rulesets*. Rulesets are generally easier to understand than trees since each rule describes a specific context associated with a class. Furthermore, a ruleset generated from a tree usually has fewer rules than the tree has leaves. However, generating rulesets requires considerably more computer time. Therefore the evaluations in the following section will be made using decision trees, although we will use selected rules from rulesets as an illustration of the knowledge of regularities acquired by machine learning. A description of a C5 rule can be found in Figure 3.

3. Evaluations

3.1. Error-rate Measuring

In the former version of AFAS, we were searching for a compromise between high *precision* (the ratio between correctly assigned functors and all automatically assigned functors) and *recall* (the ratio between the number of correctly assigned functors and the number of all functors to be assigned). Having more than half a year of experience with the previous version of AFAS, the annotators were asked whether they prefer maximizing the number of correctly assigned functors (i.e., to maximize recall) or minimizing the number of errors (i.e., to maximize precision). All of them prefer to assign automatically as many functors as possible. In this paper, we express the performance evaluation
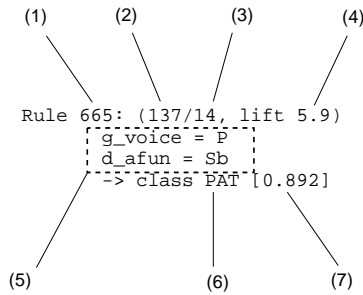
Figure 3: An example rule from the ruleset produced by C5: (1) a rule number (this is quite arbitrary and serves only for identification), (2) number of training cases covered by the rule, (3) number of cases wrongly classified by this rule, (4) the rule's estimated accuracy divided by the relative frequency of the predicted class in the training set, (5) one or more conditions that must all be satisfied if the rule is to be applicable; the conditions are of the form `attribute = value` or `attribute in {set of values}`, (6) a class (functor in our case) predicted by the rule, (7) the statistically estimated confidence of the prediction.

using only *error-rate* (number of incorrectly assigned or unassigned functors divided by the number of all functors to be assigned) instead of differentiating between precision and recall.

One way to get a reliable estimate of predictive accuracy is the *f-fold cross validation*. The data are divided into f subsets of roughly the same size and class distribution. For each subset in turn, a classifier is trained on the data in the remaining subsets and tested on the cases in the hold-out subset. The resulting error-rate is computed as an average value from the f trials. Wherever we refer to error-rate or to decision tree size in the rest of this paper (besides Table 2), we mean the average value obtained from 10-fold cross validation.

3.2. Sequence of Experiments

In order to find the importance of individual attributes for the overall performance, the system was trained repeatedly using different sets of input attributes. The experiments can be ordered into a lattice according to the subsumption of their attribute sets (Figure 4). In the following paragraphs, the experiments will be evaluated one by one and associated with an illustrative example of a rule acquired by C5. You can also see Table 1 for the evaluation summary.

Not only the sequence of experiments shows the contribution of different attributes to the performance, but it can also demonstrate the process of acquiring more "clever" rules.

E-0) No input attributes
The first experiment functions only as a baseline, because no attributes were used for decision and no decision tree was generated. Every node in the testing set was simply assigned with the most frequent functor from the training set. The resulting error rate is 88.3 %.
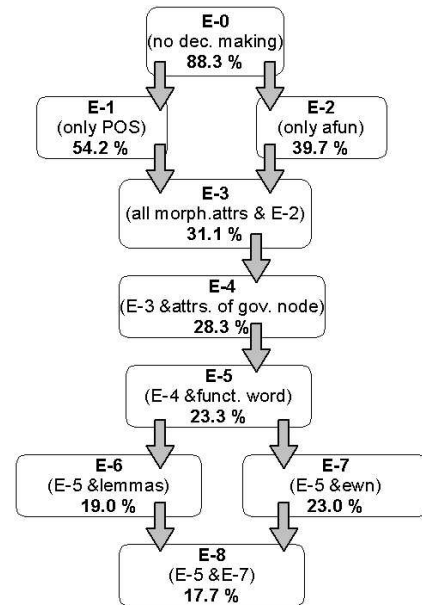

Figure 4: The lattice depicts the subsumption of attribute sets within the sequence of experiments, and the impact of enlarging the attribute set on the resulting error-rate.

E-1) Input attributes: part of speech of node N
Error rate: 54.2%
Decision tree size: 13 leaves
Sample from the ruleset:
```
Rule 7:  (3584/461, lift 4.0)
    d_pos = A
    -> class RSTR [0.871]
```
Possible interpretation of the rule: An adjective usually is the restrictive adjunct.

E-2) Input attributes: analytical function of given node.
Error rate: 39.7%
Decision tree size: 45 leaves
Sample from the ruleset:
```
Rule 21: (2244/323, lift 5.5)
    d_afun = Sb
    ->  class ACT  [0.856]
```
Interpretation: The subject of a sentence usually becomes its actor.

E-3) Input attributes: morphological attributes and analytical function of given node.
Error rate: 31.1%
Decision tree size: 416 leaves
Sample from the ruleset:
```
Rule 213: (251/130, lift 29.2)
    d_case = 3
    d_afun = Obj
    ->  class ADDR  [0.482]
```
Interpretation: An object in dative becomes addressee.

E-4) Input attributes: morphological attributes and analytical function of given node and of its autosemantic governor G
Error rate: 28.3%
Decision tree size: 1785 leaves
Sample from the ruleset:

```
Rule 388: (16/4, lift 4.7)
      g_voice = P
      d_case = 7
      d_afun = Obj
      -> class ACT [0.722]
Rule 665: (137/14, lift 5.9)
      g_voice = P
      d_afun = Sb
      -> class PAT [0.892]
```

Interpretation: The subject in a clause in passive voice becomes patient, the actor is expressed by instrumental (Compare with the rule in E-2).

E-5) Input attributes: Same attributes as in E-4, but lemmas of functional word (prepositions, conjunctions) were added.
Error rate: 23.3%
Decision tree size: 1716 leaves
Sample from the ruleset:
```
Rule 11: (63/16, lift 108.0)
      d_afun = Adv
      preposition = s
      -> class ACMP [0.738]
Rule 174: (16, lift 231.1)
      d_afun = Adv
      subord_conj = protože
      -> class CAUS [0.944]
Rule 412: (34/6, lift 368.7)
      coord_conj = nebo
      -> class DISJ [0.806]
```
Interpretations: (i) A node connected via preposition *s* ('with') represents accompaniment. (ii) A clause connected via subordinating conjunction *protože* ('because') relates to causality. (iii) A coordination node with lemma *nebo* ('or') expresses disjunction.

E-6) Input attributes: same attributes as in E-5, but lemmas of both nodes were added.
Error rate: 19,0%
Decision tree size: 5037 leaves
Sample from the ruleset:
```
Rule 511: (40, lift 28.6)
      d_lemma = rok
      preposition = v
      -> class TWHEN [0.976]
Rule 1031: (6/3, lift 3.2)
      g_lemma = činnost
      d_pos = N
      preposition = empty
      -> class ACT [0.500]
Rule 617: (11, lift 736.0)
      d_lemma = dosud
      -> class TTILL [0.923]
Rule 1397: (6, lift 71.6)
      g_lemma = patřit
      preposition = mezi
      -> class DIR3 [0.875]
```
Interpretation: (i) *v roce* ('in year') is temporal modifier. (ii) A noun directly depending on noun *činnost* ('activity') is probably actor of the activity. (iii) *dosud* ('still', 'untill now') is a temporal modifier (TTILL - 'time till ...'). (iv) *patřit mezi ...* ('belong between ...') binds a directional modifier.

Note: Rules like 1031 and 1397 can be understood as a projection of the behavior of nouns and verbs on the treebank data.

E-7) Input attributes: morphological attributes and analytical function of node N
Error rate: 23.0%
Decision tree size: 1873 leaves
Sample from the ruleset:
```
Rule 237: (115, lift 29.0)
      d_afun = Adv
      preposition = v
      d_ewn_time = yes
      -> class TWHEN [0.991]
```

Interpretation: An adverbial formed by a noun that has (according to EuroWordNet) something to do with time and that is connected via preposition *v* ('in'), is a temporal modifier of type TWHEN.

E-8) Input attributes: union of attributes from E-6 and E7
Error rate: 17.7%
Decision tree size: 4445 leaves
Sample from the ruleset:
```
Rule 70: (4, lift 132.9)
      g_lemma = množství
      d_ewn_origin = yes
      -> class MAT [0.833]
```
Interpretation: If an item depends on noun *množství* ('amount') and it is related to concept Origin in EuroWord-Net, then it has the functor MAT (material, e.g. amount of wood).

3.3. Further Observations

Experiment	Decision tree size (number of leaves)	Error rate
E-0	0	88.3 %
E-1	13	54.2 %
E-2	45	39.7 %
E-3	416	31.1 %
E-4	1785	28.3 %
E-5	1716	23.3 %
E-6	5037	19.0 %
E-7	1873	23.0 %
E-8	4445	17.7 %

Table 1: Evaluation of the sequence of experiments.

3.3.1. Robustness

As it was said above, each experiment was evaluated using 10-fold cross-validation. The question arises, how the resulting error-rate varies for the individual folds. Too large variation would imply insufficient plausibility of the acquired results and low robustness of the whole system, or in other words, high sensitivity to the choice of the testing and training set.

Table 2 shows the evaluation of experiment E-8 for each fold separately. The error-rate varies only in a narrow interval from 16.4 % to 18.1 %. This observation guarantees satisfactory robustness of the system.

Fold	Decision tree size (number of leaves)	Error rate
1	4396	16.4 %
2	4509	17.0 %
3	4509	17.7 %
4	4496	18.5 %
5	4371	17.7 %
6	4398	17.9 %
7	4381	18.9 %
8	4438	17.3 %
9	4449	17.5 %
10	4551	18.1 %
Average	4445	17.7
Std. error	18	0.2

Table 2: Evaluation of experiment E-8 on the individual folds of 10-fold cross-validation.

3.4. Error-rate versus training data size

In order to examine the dependence of the error-rate on the data set size, the experiment E-8 was repeatedly performed for data sets containing 1 %, 3 %, 6 %, 10 %, 30 %, and 60 % of all available data. The resulting values of error-rate, as well as the curve interpolated from them, are depicted in Figure 5. The curve tends to go down even in its last segment, which is promising. If we extrapolate the curve, a decrease of the error-rate by several percentage points can be expected when the size of the data set reaches 100 000 nodes. On the other hand, the "speed of the improvement" of the presented system decreases too fast, therefore it is not likely that its error-rate can ever drop under 10 % using the same attribute set and the same ML technique.

Although this observation perhaps seems to be pessimistic, it perfectly matches the fact that if two human annotators are asked to annotate (independently on each other) the same sentences, roughly 10 % of the functors differ (Hajičová et al.,2002). A part of the disagreements is naturally caused by human errors. However, there are probably also certain classes of problems, which are caused by the fact that the present annotation scheme cannot fully capture the vagueness of the language. In any case, the percentage of functors on which two people differ, makes a natural limit for what is called error rate in this paper.

4. Accelerating the Annotation Process

4.1. Annotation Environment

For the tectogrammatical annotation of the PDT, the annotators use a special Tree Editor (TrEd for short). It enables a comfortable and fast handling of the topology of the trees (correction of "wrongly-aimed" dependencies) as well as node re-labeling (changing node attributes, especially functors). More details can be found in (Hajič et
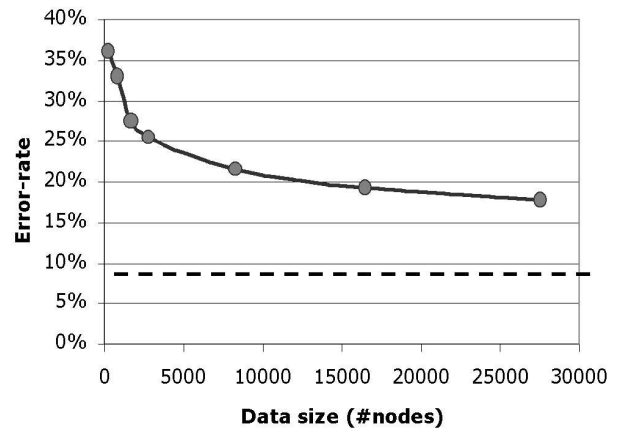

Figure 5: The improvement of the error-rate when increasing the size of the training data (the attribute set is identical with that of E-8). The data size is expressed as the number of vectors (edges). The dashed line represents the average degree of "disagreement" between two human annotators. .
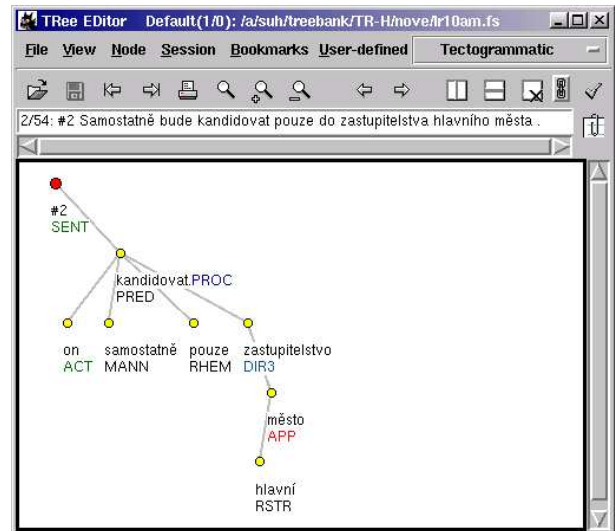

Figure 6: An illustration of the Tree Editor - the software tool for the manual annotation on the tectogrammatical level.

al.,2001). The user interface of TrEd is depicted in Figure 6.

4.2. Incorporation of the Decision Trees into the Annotation Process

In order to make the presented system for functor assignment practically usable, it has to be incorporated into the annotation process. This is done as follows: the decision tree generated by C5 is (automatically) translated into a module in Perl (the programming language TrEd is written in). In TrEd, a keyboard shortcut is defined which executes the evaluation of the decision tree subsequently for each node of the tectogrammatical tree, and it assigns functors to these nodes.

As we have seen in Figure 3, C5 estimates the confidence for every generated rule and for each leaf of the deci-
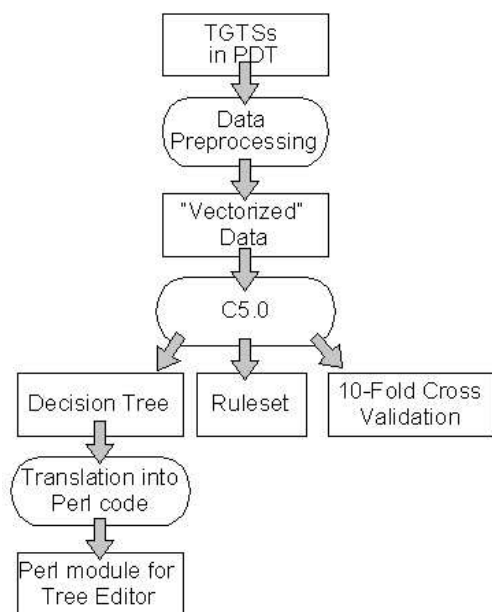
Figure 7: Data flow diagram. An the beginning, there is a set of manually annotated tectogrammatical tree structures. At the end, a Perl module for automatic functor assignment is created and can be "plugged" into the tree editor.

sion tree. We color the automatically assigned functors according to the confidence of the applied leaf of the decision tree, with the intent to make it easier for annotators to focus their concentration on the most "suspicious" nodes. Red is used for functors with estimated confidence lower than 60 %, black for confidence higher than 90 %, and blue for the rest. If an annotator corrects the automatically assigned functor, its color changes to green (the annotator can then easily perceive which functors have already been manually corrected).

4.3. Response of the Human Annotators

Currently, there are six people who have been manually annotating the tectogrammatical tree structures for at least one year. All of them have more than half a year of experience with using AFAS. They were given a questionnaire, the goal of which is to determine the real usefulness of the AFAS. Here we present selected results of this inquiry.

- All of the annotators share the opinion that the AFAS significantly increases the speed of their work. According to their subjective estimate, the manual assignment of functors would take at least twice as long without the AFAS.

- As it was already mentioned in this paper, they were asked whether they prefer to automatically assign only functors with high confidence, or to assign as many functors as possible, even though more errors would occur (precision versus recall question). All of them chose the latter possibility.

- They were also asked whether they find the coloring (expressing the confidence of functors) useful. According to their answers, it unfortunately seems that they are not using this information at all, even though

on unseen data at least 75 % of errors are among "red" functors. They check the whole tree node by node for each sentence and cannot simply skip over some parts.

- They were asked whether they are aware of any systematical error made by the AFAS. One of their suggestions actually resulted in a change of the input attribute set used by the AFAS (topological attributes have been added).

4.4. Error Analysis

Having in hand the files during the annotation of which the previous version of the AFAS was used, we can evaluate how often the automatically assigned functors were manually corrected. The resulting number of corrections (roughly 23 %) nearly exactly matches the predicted errorrate, which again confirms the robustness of this approach.

For one file (53 sentences), the errors were analyzed in detail. In this way, we try to localize the bottlenecks of the system: the most frequent types of misclassification, and (if possible) the most frequent reasons for such misclassification. Such observation are leading us to furher improvements.

5.   Conclusion

We have shown that the machine learning approach can be very helpful during the tectogrammatical annotation of the Prague Dependency Treebank, even though such annotation is very close to the semantics of natural language. Only roughly one out of five automatically assigned functors has to be manually corrected, which makes the remaining annotation significantly faster in comparison with annotation from scratch. It is also interesting that the number of differences between manually assigned functors and automatically assigned functors is less than twice as big as the number of differences between two humans.

We hope that further improvements of the presented systems will be reached via (i) larger training data, (ii) error analysis of misclassifications of the existing system, (iii) newly available lexical resources (especially a valency lexicon of Czech verbs (Lopatková,Žabokrtský,2002)).

6.   References

S. Džeroski, Z. Žabokrtský. 2001. A Machine Learning Approach to Automatic Functor Assignment in the Prague Dependency Treebank. *Prague Bulletin of Mathematical Linguistics*, 76: 35–43. Charles University, Prague.

J. Hajič. 1998. Building a syntactically annotated corpus: The Prague Dependency Treebank. In: *Issues of Valency and Meaning. Studies in Honour of Jarmila Panevová*, ed. by E. Hajičová, 106-132. Karolinum, Prague.

J. Hajič, P. Pajas, B. Hladká. 2001. The Prague Dependency Treebank: Annotation Structure and Support, In: *Proceedings of the IRCS Workshop on Linguistic Databases*, 105–114. University of Pennsylvania, Pittsburgh, PA.

E. Hajičová. 2002. Dependency based underlying-structure tagging of a very large Czech corpus. In: *Les grammaires de dpendance. Traitement Automatique des Langues 41*, ed. by Sylvain Kahane, 57-78.

E. Hajičová, P. Pajas, E. Veselá. 2002. Consistency in Deep Syntactic Annotations of Corpora: The Case of PDT. Submitted at *The 19th International Conference on Computational Linguistics*.

Hajičová, E., Panevovová, J., Sgall, P., 1999. Manual for tectogrammatical annotation. *Technical Report UFAL-TR-7*. Charles University, Prague

E. Hajičová, B. H. Partee, P. Sgall. 1998. Topic-focus, tripartite structures and cognitive content. Dordrecht: Kluwer.

M. Lopatková, Z. Žabokrtský. 2002. Valency Dictionary of Czech Verbs: Complex Tectogrammatical Annotation. In: *Proceedings of Third International Conference on Language Resources and Evaluation*. Las Palmas, Spain.

J.R. Quinlan. 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA.

Z. Žabokrtský. 2000. Automatic Functor Assignment in the Prague Dependency Treebank. In *Proceedings of the Third International Workshop on Text, Speech and Dialogue*. Springer, Berlin.

Appendix

Alphabetically Ordered List of 40 Most Frequent Functors

ACMP (accompaniement): mothers with *children*

ACT (actor): *Peter* read a letter.

ADDR (addressee): Peter gave *Mary* a book.

ADVS (adversative): He came there, *but* didn't stay long.

AIM (aim): He came there to *look* for Jane.

APP (appuerenance, i.e., possesion in a broader sense): *John's* desk

APPS (apposition): Charles the Fourth, (i.e.) *the Emperor*

ATT (attitude): They were here *willingly.*

BEN (benefactive): She made this for her *children.*

CAUS (cause): She did so since they *wanted* it.

COMPL (complement): They painted the wall *blue.*

COND (condition):If they *come* here, we'll be glad.

CONJ (conjunction): Jim *and* Jack

CPR (comparison): *taller* than Jack

CRIT (criterion): According to *Jim*, it was rainng there.

DENOM (denomination): *Chapter 5* (e.g. as a title)

DIFF (difference): taller by two *inches*

DIR1 (direction-from): He went from the *forest* to the village.

DIR2 (direction-through): He went through the *forest* to the village

DIR3 (direction-to): He went from the forest to the *village*.

DISJ (disjunction): here *or* there

DPHR (dependent part of a phraseme): in *no* way, *grammar* school

EFF (effect): We made him the *secretary*.

EXT (extent): *highly* efficient

FPHR (foreign phrase): *dolcissimo*, as they say

ID (entity): the river *Thames*

LOC (locative): in *Italy*

MANN (manner): They did it *quickly*.

MAT (material): a bottle of *milk*

MEANS (means): He wrote it by *hand*.

MOD (mod): He *certainly* has done it.

PAR (parentheses): He has, as we *know*, done it yesterday.

PAT (patient): I saw *him*.

PHR (phraseme): in no *way*, grammar *school*

PREC (preceding, particle referring to context): *therefore*, *however*

PRED (predicate): I *saw* him.

REG (regard): with regard to *George*

RHEM (rhematizer, focus sensitive particle): *only*, *even*, *also*

RSTR (restrictive adjunct): a *rich* family

THL (temporal-how-long): We were there for three *weeks*.

THO (temporal-how-often) We were there very *often*.

TWHEN (temporal-when): We were there at *noon*.