Annotation Quality Checking and Its Implications for Design of Treebank (in Building the Prague Czech-English Dependency Treebank)

Marie Mikulová and Jan Štěpánek

Charles University in Prague, Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
E-mail: {mikulova,stepanek}@ufal.mff.cuni.cz

Abstract

This article presents the system for annotation quality checking, proposed and used during the building of the Czech part of the Prague Czech-English Dependency Treebank. At first, the treebank project is introduced, as well as its basic principles and annotation process. The second part of the article pursues in detail one of the important phases of the annotation process, namely how the correctness of the annotated data is automatically and continuously checked during the process. The system of annotation quality checking is demonstrated on several particular checking procedures concerning syntactical phenomena. We try to evaluate the contribution of the system not only to the quality of the data and annotation, but also to the corpus design, impact on annotation rules and the annotation process as a whole.

1 Introduction

The annotation of a corpus is always a complex task, especially if the corpus is large and the added linguistic information is rich. The organization and management of the annotation process needs to be elaborate. If the linguistic information attached to input data (sentences) is rich and complex, the common method, multiple parallel annotation and measurement of inter-annotator agreement, becomes too costly and inapplicable for capacitive reasons. For the Prague Czech-English Dependency Treebank (PCEDT), an example of such a large complex corpus, a system of automatic quality checking of the annotated data was developed to avoid multiple parallel annotation (however, subsets of the data were still annotated in parallel).

2 The Prague Czech-English Dependency Treebank

2.1 Basic Principles

The PCEDT is planned to be a corpus of (deeply) syntactically annotated parallel texts (in English and Czech) intended chiefly for machine translation experiments. The texts for the PCEDT (its first version was described in (Čmejrek et al., 2004)) were taken from the Penn Treebank (Marcus et al., 1993), which means there are mostly economical articles from the Wall Street Journal. 2312 documents were used in the PCEDT (approximately 49,000 sentences) that are manually annotated using constituent trees in the Penn Treebank. For the Czech part of the PCEDT, the English texts were translated into Czech.

As a base of the process of creation of the corpus (hierarchical system of annotation layers, annotation rules) we use the already finished Prague Dependency Treebank (PDT) 2.0 (Hajič et al., 2006). While organizing the annotation of the PCEDT (especially its Czech part, which is the main concern of this article), we will prop ourselves upon multifarious experiences (both positive and negative) gained from the production of the PDT 2.0.

Similarly to the PDT 2.0, written sentences in the PCEDT are represented on three layers: morphological layer (lemmas, tags, morphological categories), analytical layer (surface structure, dependencies, analytical functions) and tectogrammatical layer.

The tectogrammatical layer contains all the information that is encoded in the structure of the sentence and its lexical items: the deep, semantic-syntactic structure, the functions of its parts, values of the "deep" grammatical categories, the coreference and the topic-focus articulation including the deep word order. Every sentence is represented by a tectogrammatical tree. A node of the tree either represents a semantic unit present on the surface shape of the sentence (an autosemantic word with its function words like prepositions, subordinating conjunctions, auxiliary verbs converted into various node attributes called "grammatemes") or it is a newly established node that has no counterpart on the surface—in case of ellipsis.

We adhere to the stand-off annotation principle: the layers of annotation are separated from the input data and from one another, they are interlinked by references leading always from the hierarchically higher layers to the lower ones. For example, there are two references to the analytical layer from a tectogrammatical node representing a prepositional group (as one syntactic-semantic unit): one pointing at the preposition and one at the noun.

2.2 Annotation Procedure

Tectogrammatical (deep syntactic) layer of the PCEDT that is currently being built (but also of the already accomplished PDT 2.0) represents a really broad linguistic annotation. There are 39 different attributes, for a node of a tectogrammatical tree in the PDT 2.0 there are 8.42 attributes filled on average. The annotation manual

(Mikulová et al., 2006) has more than 1000 pages.

In case of such an extensive annotation project, we find the following three aspects necessary:

- division of the annotation into several phases
- periodic measurement of inter-annotator agreement
- continuous checking of the annotation correctness

Although the annotators get their data automatically preprocessed, so that they do not have to build the syntactical tree from the very beginning (Klimeš, 2006), the annotation of the tectogrammatical layer in phases is unavoidable, because no annotator is able to keep all the annotation rules for all the annotated phenomena in her head. Moreover, the more information the annotator has to attach to the data, the more likely he or she omits some of the details.

Therefore, for the tectogrammatical annotation, we plan for the following phases:

1. building a tree structure, dealing with ellipsis included; assignment of functors and valency frames, links to lower layers (10 attributes).
2. annotation of subfunctors (fine grained classification of functors, 1 attribute).
3. annotation of coreference (4 attributes).
4. annotation of topic-focus articulation, rhematizers and deep word order (3 attributes).
5. annotation of grammatemes, final form of tectogrammatical lemmata (17 attributes).
6. annotation of remaining phenomena (quotation, named entities etc.)

The first phase is the most difficult, each annotator is responsible for the whole structure of the tree and correct values of ten attributes. All these attributes are connected with the structure and deep syntactical functions of nodes. Annotators do not have to pay attention to anything else.

To determine the amount of annotated information that does not harm the quality of the data is obviously difficult. Our belief that the current schedule of phases constitutes a reasonable and manageable rate seems to be justified by the measuring of inter-annotator agreement. The quality of the data is regularly guarded by the system of automatic checking procedures (see the following section).

3 System for Annotation Quality Checking

3.1 Motivation

A higher quality of the annotated corpus data is usually attained by parallel annotation of the same data. Particular annotated data from different annotators are mutually compared, the differences are found and manually corrected. Such a method is

very expensive (with regard to time and work) and becomes impossible if the treebank is large and the annotated information is complex. In one hour, one annotator can annotate 9.2 sentences in average in the first phase of the tectogrammatical annotation of the PCEDT. Annotation of the whole treebank (about 49,000 sentences) by one person would therefore take 5326 hours. If a person worked regularly for 20 hours a week (half-time job), the whole treebank in its first phase would take 5 years. It is clear that if we want to keep the volume of the annotated information (i.e. the annotation rate), we have to search for a different way to guard the quality of the annotated data than the usual parallel annotation.

Consequently, already during the building of the PDT 2.0, a system for automatic checking of the data was developed. The real checking took place when all the annotation had finished. The checking and fixing phase was quite complex and time-consuming; moreover, in some cases, the changes were not realized full-scale (Štěpánek, 2006). We want to avoid such a procedure in the development of PCEDT. The system of automatic checking procedures is now fully integrated into the annotation process.

3.2 Design of the Automatic Checking Procedures

Checking of the annotated data takes place continuously during the annotation process. At the beginning of the process, many automatic checking procedures were proposed and created in accordance with the annotation guidelines and more are added even during the process. Contrary to (Boyd et al., 2007) or (Dickinson, 2005), all the procedures were programmed manually, even though some of them were based on generalization of the output of an automatic inconsistency lookup.

The checking procedure proposals are based on the fact that many annotation rules imply that particular phenomena cannot (or have to) occur in the annotation output. They mainly combine attribute values and the structure of a tree.

For example, a simple check (coord002) states that every coordination has at least two members and reports all one-member (and zero-member) coordinations as errors. Another check (structure001.2) states that the root of a tectogrammatical tree has only a limited set of possible functors (PRED for a predicate, DENOM for nominative clause, PARTL for interjection clause, VOCAT for vocative clause). There is also a converse check (structure001.1) monitoring that no dependent node has the PRED or DENOM functor. The checks structure021.1 and structure021.2 guard the fact that some types of nodes, i.e. the nodes added to a tree in case of ellipsis of the governing node (such a node either has a special tectogrammatical lemma #EmpVerb or #EmpNoun or is a copy of a different node in the tree) can by definition never be leaves of a tree.

Each checking procedure has the same structure: its name consists of a category (see below) and number. A short comment explains the purpose of the procedure. All the procedures are written in Perl, as well as the main annotation tool TrEd (Pajas and Štěpánek, 2008). The output of each procedure is a table whose first column contains the name of the procedure followed by a short explanation of

```
#!btred -N -T -t PML_T -e coord()

package PML_T;

$NAME='coord002';

## Every coordination has at least two members.

sub coord {
  writeln("$NAME\tmembers\t".ThisAddress($this))
    if IsCoord($this)
       and scalar(grep $_->{is_member},$this->children) < 2;
} # coord
```

Figure 1: Checking procedure coord002

the violation of the monitored invariant. There can be any number of additional columns with any information, but the last column must contain the address of the problematic node. The annotation tool TrEd can open files with lists of addresses and can subsequently open a given tree with the cursor on a given node, and after a correction is made, it can switch to the next address in the list. An example of a checking procedure is shown in Figure 1.

The checking procedures should be accurate, i.e. they should report an error only if the data are not correct and should not report it if not. To achieve this, exceptions might be added to some of the procedures (stating something like "do not report error if the identifier of the node is T2431, the node is a leaf and its parent has functor ACT"). Several procedures had to be abandoned because of the number of exceptions getting too high.

By its name and function, each checking procedure can be classified into one of five categories:

(1) structure This category constitutes the fundamental part of the checking procedures. The procedures in this category check the structure of tectogrammatical trees, i.e. the relations between governing and dependent nodes. Particular types of nodes presume by definition only particular types of governing and/or dependent nodes. Any departure from these invariants is reported by the procedures. Besides those already mentioned, another typical example is structure027.1: it states that adjectival attribute (functor RSTR) never depends on a verb; or structure016.1, that assures that every negated verb has a #Neg child.

(2) coord The checking procedures guarding the correctness of coordination structures form a separate group. An example of the procedure coord002 was already given in Figure 1. Other procedures in this category follow from the

rules stating that some types of functors can never be coordinated together, e.g. an inner participant (argument) can be in coordination only with an argument of the same sort (coord004.4).

(3) valency The next group of checking procedures is related to valency. During the tectogrammatical annotation, each verb is assigned a valency frame from the valency lexicon describing the verb's meaning and laying down which valency modifications pertain to the given verb. (Procedure valency001.1 checks that the valency frame is assigned where required). If some of the obligatory modifications are not present on the surface, they have to be added to the tectogrammatical tree in form of so called newly established nodes. The valency checking procedures guard that all those modifications were correctly added, mainly, whether none of them is missing (valency003.2), that no modification is redundant (valency003.3), and also that the form and auxiliary words used to express the modification are listed as possible in the lexicon (valency003.4). Another checking procedure (valency004.1) monitors whether all copied verb nodes have the same valency frame assigned as their original (as in the ellipsis *John loves Mary and Peter [loves] Jane*).

(4) links The checking procedures in this category check the correctness of the links from the tectogrammatical nodes to the analytical nodes. For example, for every analytical node representing a word (i.e. not punctuation) there must be a link from a tectogrammatical tree (links001.1.1). However, the same analytical node can be linked as an auxiliary word from different tectogrammatical nodes only if the tectogrammatical nodes are coordinated in one coordination, or they or their parents have the same tectogrammatical lemma, or if all but the first tectogrammatical node are predicates with PREC children (a particle refering to a preceding context). Other procedures are based on the fact that some types of tectogrammatical nodes anticipate particular types of links, or some types of links are forbidden for them. For example, there can be no link to a preposition from a tectogrammatical node with the functor DENOM, representing a nominative clause and the same holds for a node with the functor VOCAT representing vocative clause (links003.1).

(5) attribute Checking procedures in this category do not care about the structure of a tree, only attributes of a single node are checked. All the procedures created so far focus on the tectogrammatical lemma (no other non-structural attribute is annotated in the first phase of the process). For example, attribute001.1 checks that reasons are given for every change in pre-generated tectogrammatical lemma.

Currently, there are 50 checking procedures with 103 possible violations reported (55 in the category structure, 11 coord, 15 valency, 19 links and 3 attribute).

3.3 Corrections

The checking procedures return a list of erroneous (questionable) positions in the data. The annotator gets his or her data back for corrections, manually fixing each position. As was already mentioned in the previous section, the annotation tool TrEd and the output of the procedures are compatible so that annotators only see the problematic sentences and nodes.

The checking procedures are run periodically after a given volume of the data has been annotated by an annotator (1000 sentences) or once a quarter. An error in the annotation is often reported by more than one checking procedure, each of them can report a different node affected by the error. This would confuse annotators: once they would have corrected an error in a list from one procedure, they would encounter the corrected sentence again in a different list. Moreover, by rearranging nodes in a tree an address might point to a different node than before. Therefore, the list is always filtered to contain each sentence just once in one run.

All the data (the already corrected ones included) are checked every time (in case a new check existed), and after one run of corrections, the data are checked again and again while there are any errors left (there might be some errors left by the filtering and new errors can arise in fixing the old ones).

3.4 Limits of the Automatic Quality Checking

Automatic checking procedures improve the quality of the data not only by fixing the present errors, but also by providing a feedback to the annotators (because each annotator fixes his or her own data, i.e. his or her own errors) and thus eventually improving the future annotation.

Nevertheless, even the automatic checking procedures have their limits, they cannot be used to check everything. One of the most difficult tasks is the validation of functors. The definitions of functors are very general, there are no invariants and no clues for a procedure to catch on. When assigning a functor, the annotator uses mainly his or her intuition. Of course, there is plenty of phenomena for which no automatic checking is possible. If it were possible to check the whole annotation automatically, it would be as well possible to annotate the whole data with a machine. The point of manual annotation is to obtain the data that cannot be gained automatically, because human intuition and decision of an educated annotator are needed to produce them.

However, we are aware that even the intuition of the annotators has to be revised. It is necessary to monitor whether the intuition of the annotators is roughly the same, i.e. that two different annotators will decide in (roughly) the same way in particular cases.

Consequently, some data files have been picked up for parallel annotation. The differences between annotators that are left after correcting the output of the checking procedures indicate differences in intuition, different approach of the annotators to the phenomena the checking procedures are not able to capture. We measure an

inter-annotator agreement, both for individual attributes and overall, on the files annotated in parallel. A low agreement signals that the annotation rules are too loose and we should ponder whether the rules should be made stricter.

4 Implications of the Quality Checking for the Design of the Treebank

Our system for annotation quality checking was primarily developed to search for errors in the annotated data, but its contribution for the treebank building is much broader. The checking procedures not only find errors, but also show problems related to the design of the treebank: reveal vague annotation rules, contribute to the appreciation of the annotators and in effect to the evaluation of the whole annotation procedure.

4.1 Evaluation of Annotators

Using the list of errors generated by the checking procedures, we can count how often the annotators make errors (only those errors the procedures can detect, of course). Long term regular monitoring (since the start of the annotation process) shows that the differences in error rate between annotators can be huge and that all the annotators keep their positions: no one gets markedly better nor worse (so to say, some annotators are simply more thorough than others). The comparison of veteran annotators and the new ones that annotate only for a short time is also interesting: it shows that knack, practice, and experience lead to quality of the annotation.

Table 1 shows the error rate counted on all the data annotated so far (73% of the treebank) for all the annotators individually (the number of errors made by an annotator divided by the number of sentences annotated by her), the total error rate ALL (the number of all errors found divided by the number of all the sentences annotated) and the error rate in the original data ORG (not annotated data, just automatically preprocessed; the number of sentences is lower than the number of annotated sentences because sentences annotated in parallel are counted just once, unlike in ALL).

Annotators can be evaluated by other means, too, for example by measuring the inter-annotator agreement. Such an evaluation is a part of the project as well, but it is not a subject of this article. For more details of the results of the evaluation of the project, see Mikulová and Štěpánek (2009).

4.2 Refining the Annotation Rules

The checking procedures has also lead to a refinement of the annotation rules. A condition of a checking procedure follows from (a combination of) various rules in the annotation manual, therefore one would suppose it holds everywhere in the

Annotator	Errors / Sentences	Errors per Sentence
ma	3 271 / 6 026	= 0.54
al	1 214 / 3 213	= 0.38
iv	2 648 / 8 125	= 0.33
ji	301 / 1 064	= 0.28
mi	430 / 1 786	= 0.24
ka	1 834 / 8 132	= 0.23
le	373 / 1 903	= 0.20
ol	1 177 / 6 828	= 0.17
ALL	12 139 / 39 609	= 0.31
ORG	119 090 / 34 862	= 3.42

Table 1: Number of errors per sentence

data. However, cases that are not errors are often found in the output of some checking procedures (mainly after their first run) among evident mistakes. Those are phenomena that were forgotten when formulating the rules. Thus, as a consequence of application of checking procedures to concrete natural language data, the original annotation guidelines get refined.

For example, the annotation manual states that copied verb has always the same valency frame as the original one (a verb is copied in case of actual or textual ellipses of the governing verb). The checking procedure guarding this rule was already described (valency004.1). However, the real data showed many exceptions to the rule and pointed out some errors in the valency lexicon as well. In case of an actual ellipsis of a governing verb, the manual completely omits the cases of a metaphoric, phraseological or otherwise unusual usage of the verb, as in

Na konflikt nemá dost pozornosti ani [na něj nemá] žaludek.
For a conflict, he does not have enough attention nor [he has] stomach.

"*Have attention*" and "*have stomach*" are two different meanings of the verb *to have*, thus two different valency frames—it is an exception to the rule in the manual, but also for the checking procedure. The procedure indicates inaccuracies in the valency lexicon as well when one meaning is split into several valency frames, for example *Akcie uzavřely na burze smíšeně*, *Akcie uzavřely na burze níže*, and *Akcie uzavřely na burze s mírným poklesem* (*Stocks closed mixed*, *Stocks closed lower*, *Stocks closed down modestly*). In the sentence

Akcie společnosti A uzavřely na burze smíšeně a akcie společnosti B [uzavřely] s mírným poklesem.
Company A's stock closed mixed and company B's [stock closed] down modestly.

the condition of the identity of valency frames cannot be met.

The output of the checking procedures can be compiled into a table showing the type of the most common errors. Among others, they may indicate problems in

Checking Procedure	Occurences	Percentage	Note
valency003_2_PAT_missing	883	7.27	
links001_6.1_same_aux	700	5.77	
valency003_2_ACT_missing	623	5.13	
links001_1.1_no_tnode	438	3.61	
valency001_1_no_frame	405	3.34	
valency003_4_wrong_aux	387	3.19	
structure016_1_no_neg	378	3.11	
attribute001_1_t-lemma	352	2.90	
structure003_1_fphr_lemma	348	2.87	Foreign phrase
valency003_1_invalid_lemma	345	2.84	The same lemma in the data and lexicon

Table 2: Most common types of errors. Procedures without a note are described in section 3.2.

annotation guidelines or the hardest tasks for the annotators. The ten most common errors are shown in Table 2, where the second column presents the number of occurrences and the third column presents the percentage of the given error type among all the errors found.

5 Conclusion

In this paper, we have presented our automatic system for annotation quality checking, which has been proposed and used during building the Czech part of the Prague Czech-English Dependency Treebank. The automatic checking procedures are programmed according to both the explicit annotation rules from the guidelines and the output of an automatic inconsistency lookup, and they monitor the quality of the data already during the annotation process. The output of the procedures is a list of problematic nodes that can be subsequently manually corrected.

The system for annotation quality checking was primarily developed to search for errors in the annotated data, but its contribution is much broader. The checking procedures not only report errors in the annotation, increasing the quality of the data, but also point out problematic annotation rules, contribute to the evaluation of the annotators and of the whole annotation procedure.

Acknowledgement

The research reported in this paper was supported by the LC536, GAUK 22908/2008, and FP7-ICT-2007-3-231720.

References

A. Boyd, M. Dickinson, and D. Meurers. Increasing the recall of corpus annotation error detection. In *Proceedings of TLT 2007*, volume 1, Bergen, Norway, 2007. NEALT Proceedings Series. 3.2

M. Dickinson. *Error detection and correction in annotated corpora*. PhD thesis, The Ohio State University, 2005. 3.2

J. Hajič et al. The Prague Dependency Treebank 2.0. CD-ROM, 2006. URL http://ufal.mff.cuni.cz/pdt2.0. Linguistics Data Consortium Cat. No. LDC2006T01. 2.1

V. Klimeš. *Analytical and Tectogrammatical Analysis of a Natural Language*. PhD thesis, Charles University in Prague, 2006. 2.2

M. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, (19):313–330, 1993. 2.1

M. Mikulová et al. Annotation on the tectogrammatical level in Prague Dependency Treebank. Annotation manual. Technical Report TR-2006-30, ÚFAL, Prague, 2006. 2.2

M. Mikulová and J. Štěpánek. Annotation procedure in building the Prague Czech-English Dependency Treebank. In *Proceedings of Fifth International Conference SLOVKO*, Bratislava/Smolenice, 25-27 November 2009. In print. 4.1

P. Pajas and J. Štěpánek. Recent advances in a feature-rich framework for treebank annotation. In *The 22nd International Conference on Computational Linguistics - Proceedings of the Conference*, Manchester, 2008. 3.2

M. Čmejrek et al. Prague Czech-English Dependecy Treebank: Syntactically annotated resources for machine translation. In *4th International Conference on Language Resources and Evaluation*, Lisbon, Portugal, 2004. 2.1

J. Štěpánek. *Závislostní zachycení větné struktury v anotovaném syntaktickém korpusu (nástroje pro zajištění konsistence dat) [Capturing a Sentence Structure by a Dependency Relation in an Annotated Syntactical Corpus (Tools Guaranteeing Data Consistence)]*. PhD thesis, Charles University in Prague, 2006. 3.1