Annotation Procedure in Building the Prague Czech-English Dependency Treebank

Marie Mikulová and Jan Štěpánek
Institute of Formal and Applied Linguistics, Charles University in Prague

Abstract. In this paper, we present some organizational aspects of building of a large corpus with rich linguistic annotation, while Prague Czech-English Dependency Treebank (PCEDT) serves as an example. We stress the necessity to divide the annotation process into several well planed phases. We present a system of automatic checking of the correctness of the annotation and describe several ways to measure and evaluate the annotation and annotators (inter-annotator accord, error rate and performance).

1 Introduction

Building a huge corpus with rich linguistic annotation calls for elaborate organization of the annotation process. In our contribution, we will present such a project, namely Prague Czech-English Dependency Treebank (PCEDT). In the first place, we will focus on the organizational aspects of the building of the corpus that can be generally applied to building of any similar huge corpus. In particular, the main points will be:

- division of the annotation into several phases
- system for checking the accuracy of the annotation
- ways of evaluation of the annotation and annotators

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 PCEDT were taken from Penn Treebank [4], which means there are mostly economical articles from the Wall Street Journal. 2312 documents were used in PCEDT (approximately 49,000 sentences) that are manually annotated with constituent trees in Penn Treebank. For the Czech part of 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 will use the already accomplished Prague Dependency Treebank (PDT) 2.0 [1]. While organizing the annotation of 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 PDT 2.0.

2 Division of the annotation into several phases

If one builds up a corpus in which a rich and complex linguistic information is attached to the input data (i.e. sentences), according to our experience it is advisable to divide this process into several partial phases. The question how to divide the annotation when the information attached is mostly very complex and various phenomena are interconnected remains rather difficult.

Example of such a rich annotation is the tectogrammatical (deep syntactical) layer of PCEDT (and similarly the same layer of PDT 2.0). In the annotation process, each sentence is assigned a deep syntactical structure (which among others deals with ellipsis and valency of verbs and nouns); each unit of the structure is assigned its deep syntactical function (there are several tens of the "functors") and many attributes, mainly grammatemes (tectogrammatical counterparts of morphological categories). The tectogrammatical tree captures coreference, topic-focus articulation and deep word order. There are 39 different attributes, for a node of a tectogrammatical tree in PDT 2.0 there are 8.42 attributes filled on average.

The annotation of the tectogrammatical layer was divided into several phases when the layer was being created for the PDT 2.0 already. The division is inevitable because no annotator is able to keep all the annotation rules for all the annotated phenomena in his or her head (the annotation manual [3] has more than 1000 pages). Moreover, the more information the annotator has to attach to the data, the more likely he or she omits some of the details.

The experience from production of PDT 2.0 unveiled that the division of the annotation process into several steps is desirable for the quality of the output data, even if some phenomena had to be reconsidered repeatedly by different annotators in various phases.

Today, when building PCEDT, we increased the number of phases even more. The first phase of the PDT 2.0 annotation (creating a syntactical structure and assigning functions to the units of the structure) proved to be still too complex and comprising too many features. We also tried to get rid of repeated resolution of the same problems, e.g. by introducing new temporary values for some attributes whose final values will be judged in a later phase of the annotation.

For the tectogrammatical annotation, we count on 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 still 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 the nodes. The annotator does not have to pay attention to anything else.

For the first phase of the annotation process a "working value" *#NewNode* was established for tectogrammatical lemmata of nodes added to trees in case of ellipsis of valency frame arguments. Absent obligatory arguments are represented by added nodes in final tectogrammatical trees and their tectogrammatical lemmata signify the type of the elision (*#Gen* stands for a general participant, *#PersPron* for a deletion, *#Cor* and *#QCor* for a controlee in control constructions, *#Rcp* for ellipses because of reciprocation). The type of elision is closely connected with coreference (some types of absent arguments have a coreferent, some do not). During the annotation of PDT 2.0, the lemmata of absent arguments were assigned in the first phase, the coreference in a following phase. By introducing the *#NewNode* value the final solution of the tectogrammatical lemma was postponed to the following phase together with remaining questions of coreference. An annotator inserts a node with the "working value" of the tectogrammatical lemma and only assigns its syntactical function, not taking care about the lemma.

In the first phase we also “neglect” the annotation of rhematizers (in PDT 2.0, they were annotated in the first phase). Competent decision about a rhematizer (whether an expression is a rhematizer or not, its position in the tree) is possible only if the topic-focus articulation of the sentence is decided at the same time. Therefore, definitive annotation of rhematizers is planned to the topic-focus annotation phase.

Determine an amount of annotated information that does not harm the quality of the data is obviously difficult. Our believe that the current schedule of phases constitutes a reasonable and manageable rate seems to be justified by the measuring of inter-annotator agreement (see section 4.1). The quality of the data is regularly guarded by a system of automatic checking procedures (see section 3).

3 System for Checking the Accuracy of the Annotation

When PDT 2.0 was in production, only random “manual” checks of the accuracy were performed. 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 [5].

We want to avoid such a procedure in the development of PCEDT. Checking of the data is performed in parallel with the annotation process. At the beginning of the process, a number of automatic checking procedures was proposed and new tests subsequently come up during the annotation process. Currently there are 99 checking procedures that verify the annotation of Czech sentences.

The checking procedure proposal is 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 structure of a tree. For example, a simple check states that every coordination has at least two members and reports all one-member coordinations as errors. Another check 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 etc.). There is also a converse check monitoring that no dependent node has the PRED functor, and so on.

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.

The checking procedures are run periodically after a given volume of the data has been annotated (1000 sentences) or once a quarter. All the data are checked every time (in case a new check existed) and after the correction, the data are checked again and again while there are any errors (new errors can arise in fixing the old ones).

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.

4 Ways of Evaluation of the Annotators

A system for evaluation of the annotation and annotators should by an integral part of any annotation project. In the PCEDT project, the quality of the work of a particular annotator is judged by several ways:

- the annotation agreement between annotators is measured,
- the output of the automatic checking procedures tells us how often an annotator makes mistakes compared to the others,
- the annotators book the time they spend annotating; it allows later to evaluate their performance and the relation of the efficiency to the error rate.

4.1 The annotation agreement between annotators

The basic way how to evaluate an annotation is to measure the inter-annotator agreement. However, the structure to be compared is very complex. The algorithm aligning two tectogrammatical trees built upon the same analytical tree is described in detail in [3], once the trees are aligned node to node, we just compare the values of all the attributes of all the aligned nodes. To evaluate the structural agreement, we treat the identifier of a node's parent as a new attribute of the node. Complex attributes (lists, structures etc.) need further manipulation in order to be compared. For example, identifiers of linked analytical nodes have to be sorted; for annotator's comment, we only compare the type, because the text can vary.

Since there is no “golden” annotation, we just measure the agreement of all the pairs of annotators (see Table 1, data from December 2007; average value is shown for every attribute, and average value over all the attributes and structure is presented as “Overall”). As a baseline, we use the output of an automatic procedure with which the annotators start their work (marked “Z” in the table). Note that the agreement among annotators is always higher than the agreement between any annotator and the baseline. The attributes with a lower difference between baseline and the annotators (about 5%, i.e. *is_state*, *is_generated*, *is_dsp_root*, *compl.rf*, *annot_comment,* and *a/lex.rf*) tend to contain more errors, or have too vague annotation rules.

The annotator that agrees most with all the others (“K”) is at the same time the annotator that makes the least errors and submits the most sentences (see next sections).

Overall	K	94,08%			
	Ma	94,01%			
	A	93,83%			
	O	93,78%			
	Z	84,58%			
Structure	A	88,62%	is_dsp_root	K	95,86%
	Ma	88,60%		A	95,83%
	O	87,92%		Ma	95,75%
	K	87,88%		O	95,72%
	Z	69,28%		Z	89,72%
a/aux.rf	K	93,82%	is_generated	K	96,24%
	Ma	93,58%		A	96,05%
	A	93,55%		Ma	96,03%
	O	93,53%		O	96,02%
	Z	82,45%		Z	90,27%
a/lex.rf	K	96,26%	is_member	K	94,72%
	Ma	96,12%		A	94,70%
	A	96,00%		Ma	94,50%
	O	95,90%		O	94,25%
	Z	89,67%		Z	85,47%
annot_comment	K	96,52%	is_parenthesis	Ma	95,42%
	Ma	96,40%		K	95,40%
	A	96,30%		O	95,27%
	O	96,27%		A	95,15%
	Z	90,43%		Z	88,72%
compl.rf	K	96,32%	is_state	K	96,50%
	Ma	96,22%		Ma	96,25%
	A	96,12%		O	96,13%

	O	96,03%		A	96,13%
	Z	90,18%		Z	90,35%
functor	K	85,70%	t_lemma	K	93,76%
	Ma	85,67%		Ma	93,60%
	O	85,57%		O	92,70%
	A	85,13%		A	92,42%
	Z	66,80%		Z	81,60%

Table 1: Inter-annotator agreement

4.2 Error rate

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): the number of errors the annotator made is divided by the number of sentences or nodes she annotated. Table 2 shows the comparison of the error rate for 4 annotators in December 2007 (at the beginning of the process) and current numbers for 7 annotators from July 2009. The numbers from different periods cannot be compared directly because since the beginning there have been more than 30 new checking procedures, which means the current list of errors is longer. On the other hand, the rank of the annotators can be compared.

The table shows that our current best annotator ("K") had approximately 30 errors per 100 sentences and 1.62 errors per 100 nodes. Her error rate has not got worse over the two years and she remains the best annotator. The table further shows that the differences in error rate between annotators can be great and that all the annotators keep their positions: no one gets markedly better nor worse. 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.

	December 2007		July 2009	
Who	Errors per 100 sentences	Errors per 100 nodes	Errors per 100 sentences	Errors per 100 nodes
K	29.7851	1.6241	1.5103	0.0806
O	39.6699	2.0624	4.0331	0.2067
Ma	61.4087	3.2707	8.4670	0.4533
A	63.2318	3.3498	6.3583	0.3265
L	-	-	15.0668	0.8010
Mi	-	-	16.2241	0.8460

J	-	-	19.0476	1.0971

Table 2: Error rate

4.3 Performance of the annotators

In the annotation process, even the time spent working by the annotators is measured. The annotators book the time to a web form. For each month the web application counts the annotators' performance over the month and the over-all performance. The data are important among others to determine the wages; on the basis of the data we tariff a sentence (annotators are being paid monthly according to the number of sentences they have annotated).

Table 3 shows performance of the annotators in October 2008 and June 2009, table 4 shows the over-all performance. Monitoring the performance illustrates the differences between annotators, but also the fluctuation of each particular annotator. We can also observe the inverse proportionality of the performance and error rate (see section 4.2): the more is the annotator efficient (she annotates more data), the less errors she makes.

	October 2008				June 2009			
Who	Hours	Sentences	Sentences per hour	Minutes per sentence	Hours	Sentences	Sentences per hour	Minutes per sentence
A	18.50	147	7.946	7.551	-	-	-	-
I	100.50	742	7.383	8.127	101.50	1229	12.108	4.955
J	11.50	97	8.435	7.113	2.00	28	14.000	4.286
K	33.00	418	12.667	4.737	23.50	332	14.128	4.247
L	46.00	143	3.109	19.301	27.88	365	13.092	4.583
Ma	40.00	310	7.750	7.742	-	-	-	-
Mi	17.85	142	7.955	7.542	24.91	358	14.372	4.175
O	37.81	403	10.659	5.629	56.65	632	11.156	5.378

Table 3: Performance of the annotators

Who	Hours	Sentences	Sentences per hour	Minutes per sentence
A	114.25	963	8.4289	7.1184
I	827.00	7006	8.4716	7.0825
J	105.70	1001	9.4702	6.3357

K	107.00	1430	13.3645	4.4895
L	266.41	1716	6.4412	9.3150
Ma	78.00	615	7.8846	7.6098
Mi	169.98	1655	9.7364	6.1624
O	289.02	3211	11.1100	5.4006

Table 4: Over-all performance of the annotators

5 Conclusion

In the article, we have presented some organizational aspects of building of a large syntactical treebank. We stressed mainly the necessity to divide the annotation process into several well planned phases. We presented out system for checking the correctness of the annotation. The fact that the correctness is being checked at all should be pointed out: it is not a common practice in similar projects. We described three ways to measure and evaluate the annotation and annotators.

We believe that having published PDT 2.0 with 50,000 sentences annotated on the tectogrammatical layer and being in the halftime of the PCEDT project with more than a half data already annotated (33,500 sentences, 68% of the corpus) our proposals are sufficiently backed by our experience and practice.

Acknowledgement:

The research reported in this paper was supported by the LC536, GAUK 22908/2008, and FP6-IST-5-034291-STP.

Bibliography

[1] Hajič, J. et al. (2006). The Prague Dependency Treebank 2.0. CD-ROM. Linguistics Data Consortium Cat. No. LDC2006T01. Philadelphia, PA, USA. URL:http://ldc.upenn.edu, http://ufal.mff.cuni.cz/pdt2.0.

[2] Klimeš, V. (2006). Analytical and Tectogrammatical Analysis of a Natural Language. PhD Thesis. MFF UK, Prague.

[3] Mikulová, M. et al. (2006). Annotation on the tectogrammatical level in Prague Dependency Treebank. Annotation manual. Technical report ÚFAL TR-2006-30. MFF UK, Prague.

[4] Mitchell, M. et al. (1995). Treebank 2. CD-ROM.. Linguistics Data Consortium Cat. No. LDC95T7. Philadelphia, PA, USA. URL: http://ldc.upenn.edu, http://www.cis.upenn.edu/~treebank/

[5] Štěpánek, J. (2006). 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. MFF UK, Prague.