Detecting and Correcting Errors in an English Tectogrammatical Annotation

Václav Klimeš

Charles University in Prague, Faculty of Mathematics and Physics,
Institute of Formal and Applied Linguistics
klimes@ufal.mff.cuni.cz

Abstract. We present our first experiments with detecting and correcting errors in a manual annotation of English texts, taken from the Penn Treebank, at the dependency-based tectogrammatical layer, as it is defined in the Prague Dependency Treebank. The main idea is that errors in the annotation usually result in an inconsistency, i. e. the state when a phenomenon is annotated in different ways at several places in a corpus. We describe our algorithm for detecting inconsistencies (it got positive feedback from annotators) and we present some statistics on the manually corrected data and results of a tectogrammatical analyzer which uses these data for its operation. The corrections have improved the data just slightly so far, but we outline some ways to more significant improvement.[1]

1 Introduction

Wall Street Journal collection (WSJ) is the largest subpart of the Penn Treebank ([1]). It consists of texts from the Wall Street Journal, its volume is one million words and it is syntactically annotated using constituent syntax.

The Prague Dependency Treebank (PDT), now in version 2.0 ([2]), is a long-term research project aimed at complex, linguistically motivated manual annotation of Czech texts. It was annotated at three layers: morphological, analytical, and tectogrammatical. The Functional Generative Description theory ([3]) is the main guidance for principles and rules of annotation of PDT.

In Sect. 1, we outline the tectogrammatical layer of PDT and the way of annotating the WSJ. In Sect. 2, the algorithm for detecting and correcting errors in annotation is described. An indirect method of evaluation of error corrections is given in Sect. 3 and we propose our closing remarks and possible improvements in Sect. 4.

1.1 Tectogrammatical Layer of the Prague Dependency Treebank

At the *tectogrammatical* layer of annotation ([4]), shortly *t-layer*, the sentence is represented as a rooted tree. Edges usually represent the relation of dependency between two nodes: the governor and the dependent. As some edges are of technical nature

[1] This research was supported by the Information Society projects of the Grant Agency of the Academy of Sciences of the Czech Republic No. 1ET101470416 and No. 1ET201120505.

V. Matoušek and P. Mautner (Eds.): TSD 2009, LNAI 5729, pp. 32–39, 2009.

(e. g. those capturing coordination and apposition constructions), we denote the adjacent nodes with general terms "parent" and "child".

The tectogrammatical layer captures the deep (underlying) structure of the sentence. Nodes represent only autosemantic words (i. e. words with their own meanings); synsemantic (auxiliary) words and punctuation marks can only affect values of attributes of the autosemantic words they belong to. Nodes may be created or copied, e. g. when rules of valency "dictate" to fill ellipses. Relative position of nodes at the t-layer can differ from the position of their counterparts on the surface of a sentence (if they exist). At the t-layer, as many as 39 attributes can be assigned to nodes. Some of the most important attributes follow.

- *(Deep) functor* captures the tectogrammatical function of a node relative to its parent, i. e. the type of the modification. Special functors are used for child-parent relations that are of technical character.
- The `t_lemma` attribute means "tectogrammatical lemma" and it arises from the morphological lemma of the corresponding token; when a node does not have its counterpart on the surface, the attribute has a special value indicating that it represents e. g. a general participant, or an "empty" governing verb predicate.
- For connecting with the lower layers, a link to the corresponding autosemantic word (if any) is stored in the `a/lex.rf` attribute, and links to corresponding synsemantic words (if any) are stored in the `a/aux.rf` attribute.
- In the `val_frame.rf` attribute, the identifier of the corresponding valency frame, which is kept in a separate valency lexicon, is stored.
- The `is_member` attribute states whether the node is a member of a coordination or apposition.
- The `is_generated` attribute expresses whether the node is new at the t-layer. When set, it does not necessarily mean that the node has no counterpart at lower layers – when it is a "copy" of a t-layer node, it refers to the same lower-layer node through its `a/lex.rf` attribute as the original node does.
- The `deepord` attribute stores left-to-right order of nodes at the t-layer.

1.2 Dependency Annotation of the Penn Treebank

Presently, texts from WSJ are being annotated at the t-layer ([5]), which is very similar to the Czech t-layer of PDT – the annotation guidelines are presented in [6]. In order to be able to process English constituency-annotated data, we converted and annotated the data by the respective sequence of components of *TectoMT* ([7]) – the version used is from March 2009 – an framework associating tools for language processing, in the following way.

Hand-written rules appoint the head of each phrase and the phrase structure is then converted into the dependency structure: roughly speaking, the head of a phrase becomes the parent of the other nodes in the phrase. The t-layer is created and partial tectogrammatical annotation is performed using hand-written rules: they delete synsemantic nodes and copy some information from them into the governing autosemantic nodes; they assign functors on the basis of lemmas, morphological tags and function tags; and they add new nodes in a few cases. Human annotators correct and complete

only the tectogrammatical annotation, although the annotation at the lower layers exist in the data as well.

2 Detecting and Correcting Errors

2.1 The General Idea

Some errors remain in annotation even in human-checked data, though. Guided by the effort to obtain the most correctly annotated data (from which tools using them for their training could benefit as well), we developed a procedure for searching for errors in an annotation. Inspired by [8], places where inconsistencies in annotation occur are suspected of errors, i. e. an error probably occurs if the same phenomenon is annotated in a different way at several places. The big advantage of this approach is that such places can be found easily.

While the idea presented in [8] remains, our implementation differs for the following reasons. The author of [8] developed his system primarily for checking the consistency of annotations where a label is assigned to each token in the surface representation of a text, e. g. during part-of-speech tagging. Our field of interest is a tree representation, indeed, and in this case the method has to be made more complicated. Moreover, the author stated that inconsistencies in labels of boundary tokens of his surface sequence of tokens need not denote an error, because the boundary tokens may belong to a different phrase than the rest of the sequence. This is why he had to do extra work with examining which boundary tokens are unreliable for the detection of inconsistencies.

2.2 The Way of Detecting Errors

Due to the reasons indicated above, we have chosen another approach: we do not search for tree representations belonging to a sequence of tokens; we determine the sequence of tokens for every tree structure in the data – thus no boundary tokens have to be considered. (A slight disadvantage is that the inconsistency is not revealed when the set of nodes corresponding to a sequence of tokens making one phrase does not form a tree by mistake.) When there are more different trees corresponding to a sequence of tokens, all the trees are regarded as potentially erroneous and they are written out for manual checking. It should be noted that a similar idea was used during checking the data of PDT 2.0 before their release.

A real example of the listing for the sequence *amount of $ 1 million or more* follows and Fig. 1 depicts the appropriate trees (in the same order as in the listing).

- *amount*(*or*,CONJ⟨ID⟩(*$*,⟨up⟩|*of*(*million*,RSTR(*1*,RSTR)) *more*,⟨up⟩))
- *amount*(*or*,DISJ⟨RSTR⟩(*$*,⟨up⟩|*of*(*million*,RSTR(*1*,RSTR)) *more*,⟨up⟩|*of*))
- *amount*(*or*,CONJ⟨RSTR⟩(*$*,⟨up⟩(*million*,RSTR|*of*(*1*,RSTR)) *more*,⟨up⟩))
- *amount*(*or*,CONJ⟨EXT⟩(*$*,⟨up⟩|*of*(*million*,RSTR(*1*,RSTR)) *$*,⟨up⟩(*more*,EXT)))

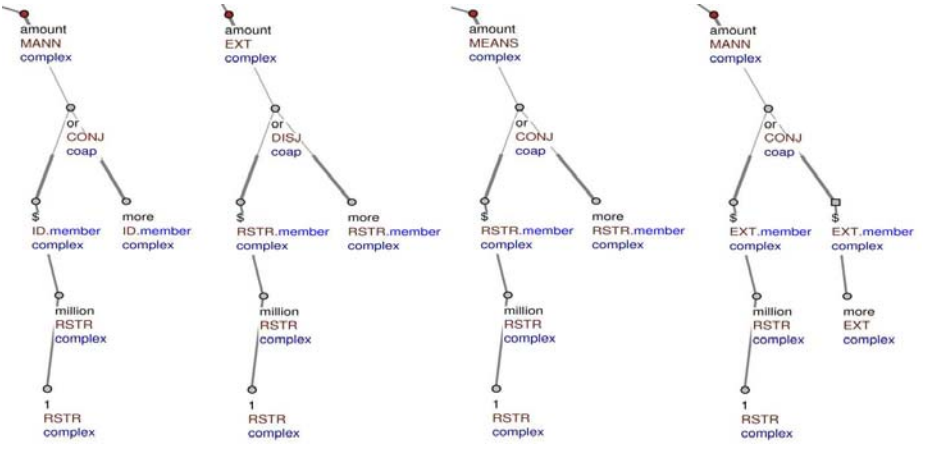

Fig. 1. Trees in the data corresponding to the sequence *amount of $ 1 million or more*

Children of a node are listed behind it in parentheses in their deep order and they are separated by spaces. Description of a node consists of its morphological lemma, a comma, its functor and optionally the '|' sign followed by morphological lemmas of its synsemantic nodes. The functor is thus the only attribute that is explicitly checked for consistency presently (although preliminary experiments with checking the consistency of other attributes were performed as well); however, we can see that the checking is affected by values of attributes `is_member`, `a/lex.rf`, `a/aux.rf`, and `deepord` as well.

The functor of the root of each tree is omitted from the listing. The reason is that it depends on the context which the tree occurs in, because the functor, expressing the relation between the root and its parent, is in fact outside the tree and not omitting it would result in generating false inconsistencies. For the same reason there are labels in angle brackets in the listing. Because of the manner of capturing the coordination and apposition constructions, functors of members of such construction have to be shifted up (denoted by ⟨up⟩ in the place of the functor) to the root of the construction (denoted by an extra functor in angle brackets), so that it could be omitted when the node is the root of the tree.

2.3 Error Corrections and Statistics

The algorithm described above was applied on the manually annotated data from WSJ containing 13,389 sentences with 228,198 nodes. 1,300 inconsistently annotated sequences were found there. Scatter of number of trees corresponding to these sequences is depicted in Fig. 2. Now we had the listing of suspected trees and we wished to correct errors in the data. (It should be noted that an idea of error corrections in syntactically annotated data was already used e. g. in [9].) Our idea was to manually choose the correct tree structure corresponding to each sequence from those found in the data. Since it was not possible to assign correct trees to such a high number of sequences, only those

sequences where the judgement can help the most were selected in the following manner. Total number of occurrences minus the number of occurrences of the most frequent corresponding tree was assigned to each sequence. This quantity expresses the minimum number of errors in consistency if we hypothesize that only one tree is correct for each sequence in any context. For 331 sequences where this quantity was greater than 1, the correct tree was being chosen from those occurring; and for as few as 152 of them it really was chosen. The reasons why no correct tree could be chosen are that either it was not present among the occurring ones, or the annotation rules could not make a decision.[2] For 37 of these 152 sequences, the correct tree was not the most frequent one.

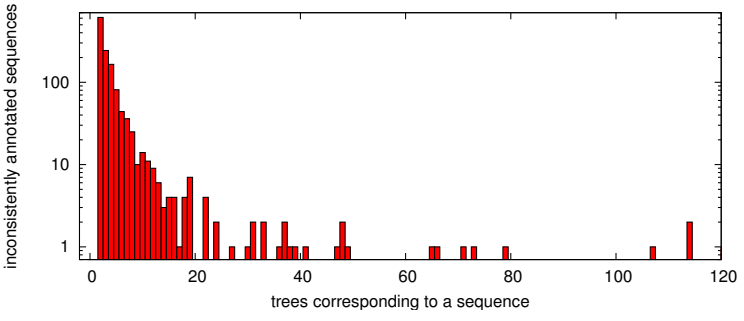

Fig. 2. Numbers of inconsistently annotated sequences depending on numbers of trees corresponding to a sequence. One sequence with 297 corresponding trees is omitted for the sake of clarity of the graph.

Because the trees corresponding to mere 26 of 152 sequences differ in structure (trees corresponding to the rest of sequences differ in functors only), we decided to implement the correction of functors only in the first stage. In the data, 2,371 occurrences of inconsistently annotated sequences were found and 843 corrections of a functor were made (0.0037 corrections per node).

In order to estimate the potential of the corrections, the most frequent tree corresponding to each sequence having no correct tree assigned was denoted as correct. The corrections were made once again: there were 6,351 occurrences of inconsistently annotated sequences in the data and 2,396 corrections of a functor were made (0.0105 corrections per node). We remark that the potential of the method is significantly higher because it was the most frequent tree which was considered correct, and thus the number of corrections was the lowest possible.

The most frequent inconsistently annotated sequences with numbers of corresponding trees of particular types are *New York*: 281+10, *stock market*: 53+52+9, *last year*: 87+26, *Wall Street*: 92+13+1, *big board*: 77+2, and *Dow Jones*: 38+35.

[2] This holds true for valency of nouns mainly – the annotation rules are not complete yet; moreover, they are still in progress.

3 Indirect Evaluation through TBLa2t

In order to be able to evaluate how the data improved by the corrections of inconsistencies, we used them for training and testing of a tectogrammatical analyzer called *TBLa2t*.

3.1 What Is TBLa2t

TBLa2t is developed for the tectogrammatical annotation of PDT-like annotated texts in virtually any language, given it (e. g.) the partial tectogrammatical annotation described in Subsect. 1.2. It is based on machine-learning method (to be specific, on transformation-based learning), which means that it uses training data for creation of its language model. It operates in several phases and it is able to create, copy and delete nodes, to assign values of attributes and to perform complex transformations of tree structures. Its English clone is described in [10].

3.2 Evaluation

Since two tectogrammatical trees constructed over the same sentence do not necessarily contain the same number of nodes, a node from a tree from the test annotation is paired with a node from the corresponding tree from the correct annotation in the first step of evaluation and each node is a part of at most one such pair. Only after performing this step, attributes of the paired nodes can be compared.

We define *precision* in an attribute to be the number of pairs where both nodes have a matching value of the attribute divided by the total number of nodes in the test annotation; and we define *recall* to be the same number divided by the total number of nodes in the correct annotation. For our results, we report *F-measure*: it is the equally weighted harmonic mean of precision and recall.

When we want to compare the structure, we have to modify this approach. We define a node to be correctly placed if the node and its parent are in a pair and the counterpart of the parent of the node in question is the parent of the counterpart of the node in question.[3] Then the numerator of the fractional counts from the paragraph above is the number of correctly placed nodes.

The `a/aux.rf` attribute is a set attribute, and thus it requires special treatment. We evaluate it en bloc and consider it to match if the whole set of links matches its counterpart.

3.3 Results

The table below summarizes the F-measure of results of our experiments with TBLa2t. For training, we used 10,994 sentences with 187,985 nodes; for testing, we used 1,116 sentences with 18,689 nodes.

In the first and second set of columns, the results obtained on the data before and after processing by TBLa2t are captured, respectively. The first column of each set describes

[3] Informally, the “same” node has to depend on the “same” parent in both trees.

the data without corrections of inconsistencies; the second column is related to the data with manually made corrections; the third one is related to data where the most frequent tree structure was considered correct when human judgement was missing (see Subsect. 2.3).

	Before TBLa2t			After TBLa2t		
attribute	**without**	**manual**	**all**	**without**	**manual**	**all**
structure	82.6%			87.5%		
functor	56.6%	56.4%	56.5%	79.3%	79.5%	79.6%
`val_frame.rf`	87.3%			93.2%		
`t_lemma`	86.0%			90.0%		
`a/lex.rf`	93.6%			96.4%	96.4%	96.5%
`a/aux.rf`	78.7%			81.2%		
`is_member`	90.7%			93.8%		
`is_generated`	93.0%			95.2%		

Fig. 3. The F-measure of structure and attributes in the data before and after processing by TBLa2t – without corrections of annotation inconsistencies, with manual corrections, and with all corrections

We can see that the results differ only slightly: for manual corrections we got 1.0% error reduction in functors. We might expect better results for data with all corrections; however, these were definitely made worse by the way the judgements were made: for similar cases the choice of the "correct" tree structure might have been done inconsistently, and thus it could not improve the data.

As a side effect, we got the best results for tectogrammatical analysis of English: in [10], there was reported F-measure of 85.0% for structure and 71.9% for functors. The reasons for improvement are (in estimated descending order of importance) higher amount of data for training, different guidelines of annotation, and improvements of the code of the analyzer.

4 Closing Remarks and Possible Improvements

We introduced a working method for detecting and correcting inconsistencies in the human-revised tectogrammatical annotation. The listing of inconsistencies got very positive feedback from annotators and its importance will grow together with the amount of annotated data, because new sequences will be less frequent.

The benefit of corrections is modest; however, it will rise as the annotation guidelines will be made clearer and more complete, because for more sequences, one of the trees will be able to be denoted correct – and listing of inconsistencies can help with this significantly, since it points out the annotation rules which are not clear. By the time the number of inconsistencies will be relatively small, other ideas from [8] will be possible to employ, e. g. automatic selection of the correct tree from those existing (it often does not exist presently).

Another idea is to implement regular expressions for the description of particular tokens of sequences: when manually choosing the correct trees, very little amount of extra work could result in much higher coverage.

Even the presentation of inconsistencies can be improved, e. g. for the sake of clarity, those subtrees of trees corresponding to a sequence can be omitted from the listing, which no difference exists in. However, such decisions should always emerge from the needs of annotators.

References

1. Marcus, M.P., Santorini, B., Marcinkiewicz, M.A.: Treebank-2, Linguistic Data Consortium, Philadelphia, LDC Catalog No. LDC95T7 (1995), http://www.cis.upenn.edu/~treebank/
2. Hajič, J., et al.: Prague Dependency Treebank 2.0, Linguistic Data Consortium, Philadelphia, LDC Catalog No. LDC2006T01 (2006), http://ufal.mff.cuni.cz/pdt2.0/
3. Sgall, P., Hajičová, E., Panevová, J.: The Meaning of a Sentence in Its Semantic and Pragmatic Aspects Academia. Kluwer, Praha (1986)
4. Mikulová, M., et al.: Annotation on the tectogrammatical level in the Prague Dependency Treebank, ÚFAL Technical Report no. 2007/3.1, Charles University in Prague, Prague (2007)
5. Hajič, J., et al.: Prague English Dependency Treebank 1.0, Charles University in Prague, Prague (2009) (CD-ROM) ISBN 978-80-904175-0-2
6. Cinková, S., et al.: Annotation of English on the Tectogrammatical Level. ÚFAL Technical Report No. TR-2006-35, Charles University in Prague, Prague (2006)
7. Žabokrtský, Z., Ptáček, J., Pajas, P.: TectoMT: Highly Modular MT System with Tectogrammatics Used as Transfer Layer. In: ACL 2008 WMT: Proceedings of the Third Workshop on Statistical Machine Translation, Association for Computational Linguistics, Columbus, Ohio, pp. 167–170 (2008)
8. Dickinson, M.: Error detection and correction in annotated corpora. PhD thesis. The Ohio State University (2006), http://ling.osu.edu/~dickinso/papers/diss/
9. Štěpánek, J.: Závislostní zachycení větné struktury v anotovaném syntaktickém korpusu (nástroje pro zajištění konzistence 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)
10. Klimeš, V.: Transformation-Based Tectogrammatical Dependency Analysis of English. In: Matoušek, V., Mautner, P. (eds.) Proceedings of Text, Speech and Dialogue 2007, pp. 15–22. Springer Science+Business Media Deutschland GmbH, Heidelberg (2007)