Comparing Conversions of Discontinuity in PCFG Parsing

Yu-Yin Hsu

Indiana University
Bloomington, IN, USA
E-mail: hsuy@indiana.edu

Abstract

This paper compares three different types of representations that resolve long-distance dependency into binary representation as it is required in PCFG parsing. Each conversion is applied to the German TIGER Treebank in the PCFG parsing experiments. The examination of data and the labeled dependency evaluation show that the choice of conversion of treebank data in the preprocessing step can influence the F-score up to 2.83% and that each conversion has its own advantages and limits. The result of this paper shows that this preprocessing step is not trivial in parsing free word order languages.

1 Introduction

In order to conduct a Probabilistic Context Free Grammar (hereafter PCFG) parsing, it is necessary to resolve crossing branches in the input data, because such parsers only process context-free tree structures, where no long distance relationship is allowed [11]. In the literature, crossing branches are mostly resolved by a node-raising approach (see approaches by Kübler [4]; Maier [6]; Kübler et al. [5]). Boyd [1] proposes a new approach: a node-splitting approach, i.e., mother nodes that are associated with discontinuous daughter nodes are splitted into partial nodes in order to resolve the discontinuity. She argues that such a node-splitting representation is better than the node-raising method, because the converted representation retains the original syntactic information after resolving crossing branches and thus structures are recoverable. However, no work discusses the impact of this conversion step on the parsing performance so far. Nonetheless, this preprocessing step is not trivial, given that German allows free word order and that about 30% of the sentences in the TIGER Treebanks has one or more than one crossing branch. Thus, in this paper, I focus on how different modifications of the TIGER Treebank data affect the parsing results. In addition to the aforementioned two approaches, a new approach is also considered, the node-adding approach. This method modifies the tree structure by copying the mother node information during the conversion.

The remainder of the paper is organized as follows. In section 2, I briefly introduce TIGER Treebank. In section 3, I summarize the preprocessing methods used in the experiments, and in section 4, I show the three different representations at issue. Section 5 explains the experiment followed by evaluation and discussion. Section 6 concludes the paper.

2 TIGER Treebank

The TIGER Treebank Release 2 contains 50 474 sentences from the German newspaper *Frankfurter Rundschau*. Part-of-speech, lemma and morphological information, phrase structures and grammatical function are annotated in the treebank. TIGER Treebank can be searched *via* TIGERSearch, which returns tree diagrams of query structures.

Among the four German treebanks that are available, NEGRA [10], TIGER [3], and the Tübingen Treebank of Written German (TüBa-D/Z) [12] are treebanks of written data. All three of them use the Stuttgart-Tübingen-Tagset [8] for part-of-speech annotation. TIGER, like NEGRA yet unlike TüBa-D/Z, does not allow unary branching and its tree structures are flatter, whereas TüBa-D/Z presents more hierarchy and it includes topological fields to avoid long distance dependencies and thus no crossing branches occur in TüBa-D/Z.

In TIGER, subjects and finite verbs are always treated as immediate daughters of the clauses, while non-finite verbs, complements of the verb, PPs and adjuncts are daughters under a single VP. It is also common to have topicalization or extraposition in German sentences. Thus, crossing branches are used in TIGER to account for such long-distance dependencies in German. Sentence structures in TIGER show less hierarchy in order to avoid structural ambiguities and to eliminate the need for traces. The distinction between adjuncts and arguments is shown through labels of syntactic functions, rather than in the structure [3]. Figure 1 illustrates an example of TIGER trees with a discontinuous constituent, i.e., the VP *Bis dahin geheimgehalten* 'until then kept secret'. Therefore, it is necessary to resolve crossing branches by a conversion of the data representation before doing PCFG parsing.

3 Data Preparation

Several preprocessing steps were done for the TIGER treebank data in order to prepare appropriate input files that meet the requirements of the parser. LoPar [9] was used in the experiments. It is an implementation of a parser for head-lexicalised probabilistic context-free grammars. Among 50 474 sentences in TIGER treebank, sentences with a length of more than 40 words were excluded in the experiments, because they cause problems with memory load in the parsing process. 90% of the filtered data was used as training data, 5% as development data and 5% as test data.

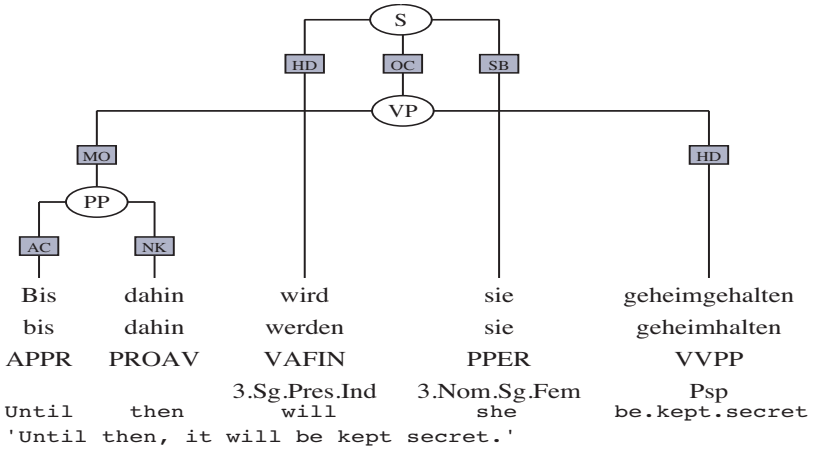

Figure 1: An example of TIGER tree

POS tags along with their labels of grammatical function were extracted directly from the treebank. Because this project focuses on the effect of different representations of crossing branches, gold standard labels were used to avoid unnecessary noise introduced by automatic POS tagging.

Next, three converted sets of treebank data, i.e., data with node-raising, node-splitting and node-adding, were created based on the same 90-5-5 split. In each conversion, a virtual root node was assigned to each sentence, and punctuations were re-attached to their local surrounding nodes.

4 Three Different Representations

4.1 Node-raising Approach

Following the node-raising approach [4] [5] [6], a script was used to detect crossing branches and to resolve crossing branches by raising non-head sister(s) higher up until no crossing branches were observed in the tree. After the raising conversion, the sentence in Figure 1 is shown in Figure 2. I.e. the prepositional phrase PP, which is the non-head daughter, is raised from the VP to the S node.

The raising approach is good in maintaining the number of nodes involved in a tree, i.e., both Figure 1 and Figure 2 have three phrases: S, VP, PP. However, after raising, the PP is reattached to a new, higher node; the mother node information (i.e., the mother node of PP-MO is VP-OC) is not available in the new structure in Figure 2. This leads to a new rule in the grammar: S-> PP VAFIN PPER VP, which did not occur before.

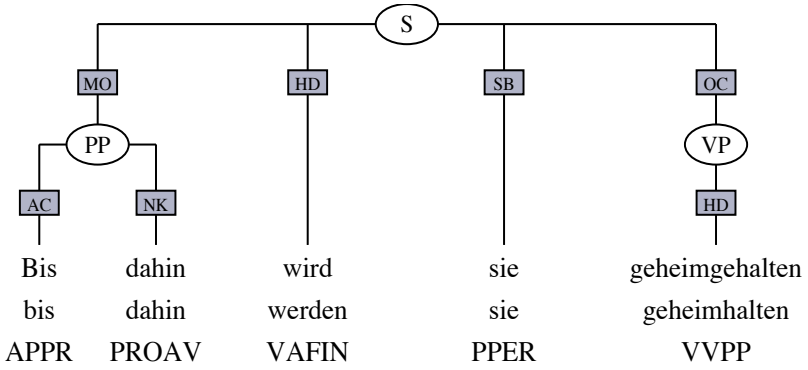

Figure 2: Node-raising conversion

4.2 Node-splitting Approach

Unlike the node-raising approach, Boyd [1] suggests a node-splitting representation to resolve crossing branches. That is, the mother nodes of discontinuous daughter nodes are divided into partial nodes marked with an asterisk. She argues that such representation is easily reversible and the original structural information can be maintained in this conversion. I replicated her idea of conversion; a script was used to detect long distance constituents and to split nodes that involve crossing branches. For the sentence in Figure 1, it is converted into the tree in Figure 3 after the splitting conversion. We can see that the node VP splits into two VP*s in Figure 3.

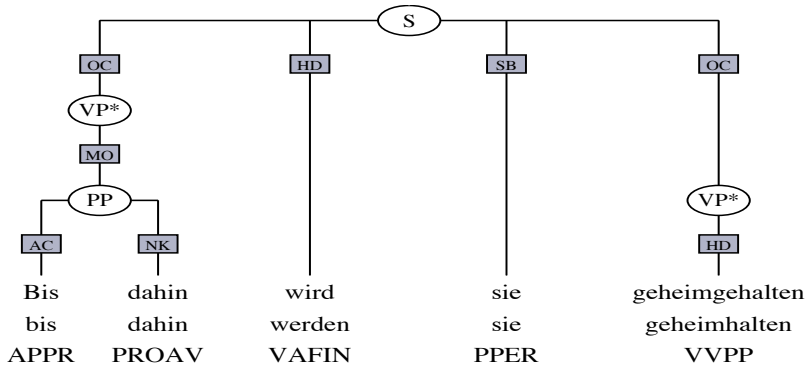

Figure 3: Node-splitting conversion

Boyd [1] also notes that in this node-splitting representation, it is easy to recover the original structure; however, if there are discontinuous nodes that use the same labels, or if there are nodes of the same categories that were split more than once, it is more difficult to find the right pair of split nodes in parsing. Figure 4 shows a parsed output that has four split nodes of VP and it would be more challenging for a program to identify which two VP*s should be combined together in order to recover the original structure. Fortunately, nodes like these can never

be sisters in most of the data and thus such conversion is still reversible. An additional problem is that parsers make grouping decisions in individual processes, which means that there is no guarantee that there is always a second starred node.

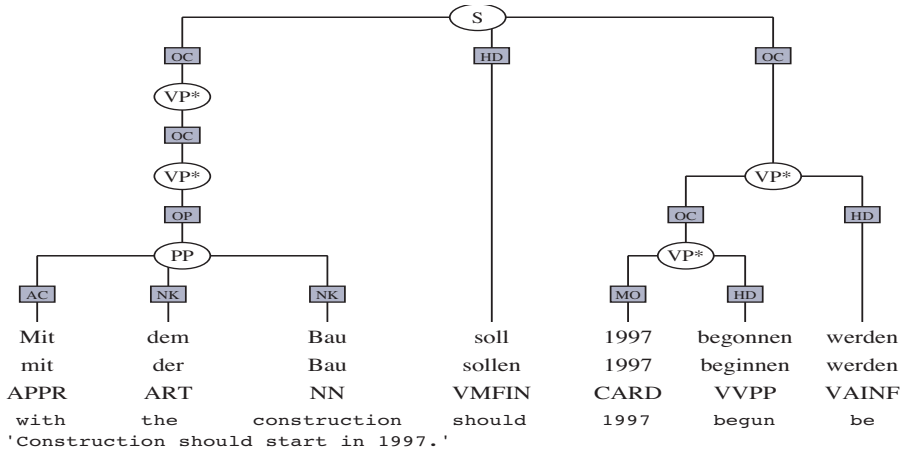

Figure 4: A tree with more than one splitting node of the same label

4.3 Node-adding Approach

In addition to raising and splitting approaches, another possibility that has not yet been used is to resolve crossing branches by copying the mother node information and keep such information through additional nodes in the conversion. A script was used to detect crossing branches, duplicate their mother node information, and then linked the new copy of mother node with its non-head daughter to resolve crossing branches. A similar process continued until no discontinuous constituents were found in the structure. After this conversion, the sentence in Figure 1 is represented as in Figure 5.

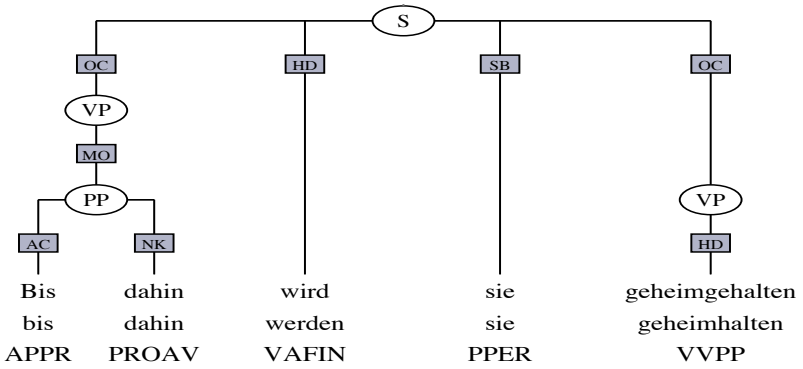

Figure 5: Node-adding conversion

This conversion as well as all other conversions introduces inconsistencies in the grammar, because new and specific unary nodes are introduced into the grammar. However, this method produces fewer inconsistencies than the node-splitting approach because it does not introduce new node labels.

5 Experiment

5.1 Results

In the LoPar parsing experiments of the TIGER Treebank data, the *viterbi* function was used to return only the best analysis of each test sentence. Given the *viterbi* output of LoPar of each conversion, a series of scripts were used to perform the PARSEVAL measures. The evaluation was computed relative to the number of brackets in a sentence structure and then returned precision, recall and F-score for labeled constituents. Overall, 28% of the test sentences involved crossing branches, and 86% of them had two crossing brackets or fewer. Table 1 shows the results of three different conversions at issue. We can see that the node-adding approach shows a precision slightly higher than the raising approach, but higher than the node-splitting approach by about 3%. In other words, both the node-raising and the node-adding approaches returned more precise parses than the node-splitting approach. In terms of recall, the raising approach is 1.12% higher than the node-adding version and it is 1.74% higher than the splitting version. Such differences can be because the new node labels introduced by the node-splitting approach cannot be estimated reliably by the parser, while the node-adding approach avoids such a problem and therefore reaches the highest precision of all methods (In section 5.2, I further discuss the effect of different representations of node labels).[1]

	Precision	Recall	F-score
Raise	60.00	64.49	62.16
Split	57.59	62.75	60.06
Add	60.74	63.37	62.03

Table 1: PARSEVAL results (%)

	Precision	Recall	F-score
Raise	52.94	57.73	55.23
Split	51.32	53.52	52.40
Add	54.21	53.37	53.78

Table 2: PARSEVAL results of sentences with crossing branches(%)

The same evaluation process then was carried out on only sentences with crossing branches (i.e., 28% of the test sentences). The results are shown in Table 2.

[1] As it is pointed out by a reviewer, we acknowledge that the standard evaluation measures are not perfect and the measure used in this study is specific to one type of treebank formats. The impact on changing the number of nodes in the structure is discussed in section 5.2. It is possible to include other types of measures, such as a dependency evaluation (as discussed by Boyd and Meurers [2]). However, such measures rely on extracting information from gold standard as well as the parser output. Rehbein and van Genabith [7] showed that the conversion from gold standard constituents is very reliable, but it is unclear how reliable it is for parser output, which may contain unexpected structures that will lead to conversion errors. Thus, we leave the comparison of different types of evaluation measures for future work.

Similar to the overall results, the node-adding version shows the highest precision and the raising approach shows the best recall among the three conversions. However, the node-splitting conversion shows a recall slightly higher than the node-adding conversion, but it is much worse than the node-adding conversion in precision. Results are all lower than 60% and the node-raising version shows the best F-score, 55.23%, which is 1.45% higher than the node-adding version, and is 2.83% higher than the node-splitting version. The results show that the choice of conversion in the preprocessing affect the parsing of sentences with crossing branches and the difference could be up to 2.83%. For conversions that preserve original structural information, the node-adding approach shows an F-core about 2% higher than that of the node-splitting approach. In sum, both node-raising and node-adding conversions show better F-score than the node-splitting representation.

5.2 Rule Types

The lower scores of the splitting version may be because of the asterisk shown on labels that leads to more unique rules and affects the frequency of rules, and in turn influence the parsing results. I investigated this assumption by looking at the number of rules extracted form the training data by the different methods. The results reported in Table 3. Taking the raising approach as the baseline, we can see that the node-splitting approach has the largest set of phrase structure rules (7.9% more than the raising approach), which is higher than the number of rules created by the node-adding approach (6.7% more than the raising version).

Raise	Split	Add
261682	282315	279293
	+7.9%	+ 6.7%

Table 3: The number of phrase structure rules in the training data

These differences also reflect on the numbers of rule types for major phrase types. As shown in Table 4, although the exact rules in each set of representation are not identical, in terms of the number of rule types of the major categories, the raising and adding-node approaches have similar numbers of rule types of these categories, but splitting-node approach creates additional rules. Numbers in parentheses indicate the number of rules having partial nodes. Node-splitting conversion increases rule types of each category, and this change also affects the frequency of rules. Table 5 shows the top five frequent rule types in each set of data. Although both node-splitting and node-adding conversions increase the number of nodes in the modified structure, the node-adding approach does not change the ranking of rule frequency much, but the node-splitting approach shows more effects on the frequency of rules and the parsing performance.

	PP	VP	S
Raise	24	16	18
Split	32 (8)	21 (5)	29 (11)
Add	24	16	18
	AP	AVP	NP
Raise	20	22	29
Split	27 (7)	29 (8)	46 (17)
Add	20	22	29

Table 4: Rule types

Frequency Rank	Split Rule Type	Add Rule Type	Raise Rule Type
1	PP-MO	VP-OC	PP-MO
2	NP-SB	PP-MO	VP-OC
3	VP-OC	NP-SB	NP-SB
4	VP*-OC	NP-OA	PP-MNR
5	PP-MNR	PP-MNR	NP-OA

Table 5: Top five rule types in the frequency rank

Train		Test	
Phrase	GR	Phrase	GR
VP 50.5	OC 51.9	VP 49.6	OC 51.6
PP 16.3	MO 15.1	NP 17.3	MO 14.9
NP 15.5	SB 8	PP 15.8	SB 8

Table 6: The most frequently modified nodes (%)

5.3 Modified Nodes

Boyd [1] reports that in the splitting conversion, VP (about 55%) and NP (about 20%) are the most frequently split nodes. In this project, a similar phenomenon is found. Table 6 summarizes the most frequently modified phrase types and the grammatical functions (GR). In both train and test data, VP is the most frequently modified category (about 50%); PP and NP come as the second and the third frequent categories. In terms of the grammatical function, clausal objects (OC) are those that underwent modification most frequently (about 52%), and then modifiers (MO) come as the second (about 15%), and the subject (SB) the third. Among nodes being affected in the conversion process, words with labels of VP-OC, PP-MO, NP-OA and PP-OP are the most common nodes that underwent changes. This, in fact, reflects the linguistic properties of German that VPs often involve long-distance dependency, and that PPs and objects often involve extraposition, and thus they are the targets in the conversion process more frequently.

5.4 Errors in the Parses

In the parsing output, the most common errors are found with PP modifiers, clausal VP objects, subjects and objects. PPs that are nominal modifiers were often parsed as general PPs, and a general PP might be identified as nominal modifiers sometimes. NP errors were found mostly when the object precedes other NPs in the sentence and was parsed as the subject. Constituent errors happened mostly with the VP clausal object. These errors are not easily avoidable, since the function *viterbi* is calculated based on the probability of rules, and VP-OC rules have a dominating frequency in the data, comparing to other VP rules or clausal rules (cf.

Table 5). Errors of PP adjunction are also predictable based on the frequency of rules, i.e., PP-MO rules have a frequency higher than PP-MNR in all three versions. In addition to these general parsing errors, the node-splitting conversion shows a different problem. Since *viterbi* is calculated based on the probability of rules, the parser does not know that it is necessary to match partial nodes into a complete node in parsing. Therefore, some parses of the splitting version show more partial nodes than necessary. Figures 6 to 8 demonstrate this problem. Figure 6 is one of the sentences in TIGER that involves crossing branches, i.e., the VP has a PP at the beginning of the sentence and the past participle *übriggeblieben* 'left over' at the end of the sentence. Ideally, the crossing branches in this sentence would be resolved through the node-splitting conversion, as in Figure 7, where both the initial PP and the final participle *übriggeblieben* show the same mother node information, i.e., VP*-OC.

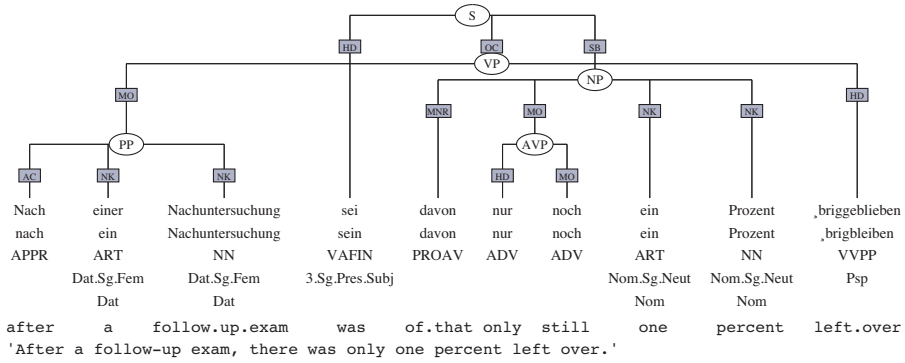

Figure 6: An example of original TIGER trees

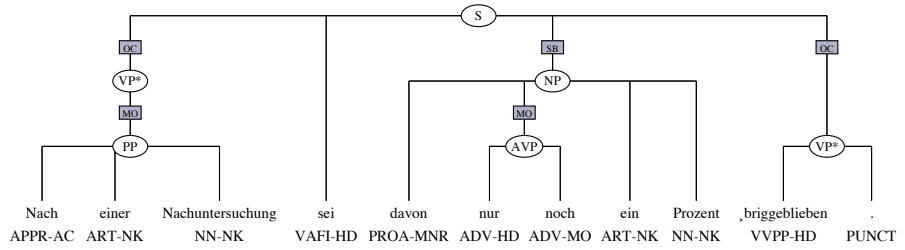

Figure 7: The expected splitting modification of Figure 6

However, since the probabilistic calculation of the best parse is independent of the structure, the parser does not know it is important to match partial nodes in the parsing. For the sentence in Figure 6, the parser returned the structure in Figure 8. We can see that it returned more smaller constituents (and this tendency is less observable in the other two approaches). In addition, a partial node, NP*-PD, for the word *davon* 'of that' occurs in Figure 8, but there is no corresponding NP*-PD

partial node in the structure.

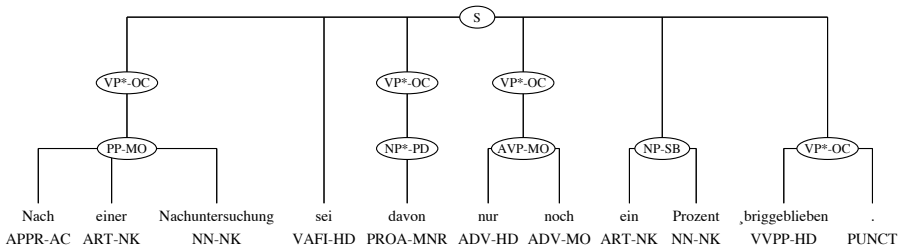

Figure 8: The actual node-splitting parse of the sentence in Figure 6

6 Conclusion and Future Work

The results of the experiments show that the choice of a conversion algorithm influences the parsing results by up to 3%. The experiment reports a 1-2% difference in recall, a 3% difference in precision and overall a 2% difference in the F-score when a different modification process is chosen. Although both the node-adding and the node-splitting conversions try to maintain the original information in the structure, the node-spitting version is recoverable, but it is harder to recover the original trees from the node-adding conversion, since there is no indication showing which nodes should be combined together. In terms of recoverability, the node-splitting approach is better than the other two modifications, but in maintaining the original structural information and in terms of the parsing performance, the node-adding approach seems to be preferred. To further improve the parsing results, we may consider adding morphological information. This extra information would be most helpful to distinguish object NP from subject NP in parsing, and avoid the bias introduced by linear order of words. Theoretically, this can also help identify PP adjunctions. I leave these possibilities for the future work.

Acknowledgements

I am grateful to Sandra Kübler for her insightful suggestions, comments and support for this project. I would also like to thank the anonymous reviewers for their valuable comments.

References

[1] Boyd, Adriane (2007) Discontinuity revisited: An Improved Conversion to Context-free Representations. In *Proceedings of the Linguistic Annotation Workshop (LAW 2007)*, Prague, Czech Republic.

[2] Boyd, Adriane and Detmar Meurers (2008) Revisiting the Impact of Different Annotation Schemes on PCFG Parsing: A Grammatical Dependency Evaluation. In *Proceedings of the ACL'08 Workshop on Parsing German*, Columbus, Ohio.

[3] Brants, Sabine, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith (2002) The TIGER Treebank. In *Proceedings of the First Workshop on Treebank and Linguistic Theories (TLT 2002)*, Sozopol, Bulgaria.

[4] Kübler, Sandra (2005) How Do Treebank Annotation Schemes Influence Parsing Results? Or How Not to Compare Apples and Oranges. In *Proceedings of the Fifth International Conference on Recent Advances in Natural Language Processing (RANLP 2005)*, pages 293–300, Borovets, Bulgaria.

[5] Kübler, Sandra, Erhard W. Hinrichs, and Wolfgang Maier (2006) Is It Really That Difficult to Parse German? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, Sydney, Australia.

[6] Maier, Wolfgang (2006) Annotation Schemes and Their Influence on Parsing Results. In *Proceed- ings of the ACL-2006 Student Research Workshop*, Sydney, Australia.

[7] Rehbein, Ines and Josef van Genabith (2007) Treebank Annotation Schemes and Parser Evaluation for German. *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, Prague, Czech Republic.

[8] Schiller, Anne, Simone Teufel, and Christine Thielen (1995) *Guidelines fü das Tagging deutscher Textkorpora mit STTS*. Universität Stuttgart, Universität Tübingen, Germany.

[9] Schmid, Helmut (2000) Lopar: Design and Implementation. Technical report, Universität Stuttgart, Germany.

[10] Skut, Wojciech, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit (1997) An Annotation Scheme for Free Word Order Languages. In *Proceedings of the Fifth conference on Applied Natural Language Processing (ANLP)*, Washington, D.C.

[11] Telljohann, Heike, Erhard W. Hinrichs, and Sandra Kübler (2004) The TüBa-D/Z treebank: Annotating German with a Context-free Backbone. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, pages 2229–2235, Lisbon, Portugal.

[12] Telljohann, Heike, Erhard W. Hinrichs, Sandra Kübler, and Heike Zinsmeister (2005) *Stylebook for the Tübingen Treebank of Written German (TüBa-D/Z)*. Universität Tübingen, Germany.