Directed Treebank Refinement for PCFG Parsing

Tylman Ule

Seminar für Sprachwissenschaft, Universität Tübingen
ule@sfs.uni-tuebingen.de

1 Introduction

The linguistic annotation of a treebank conforms to an annotation scheme, which has to serve several purposes. It should be easy to understand and follow by human annotators, and it should naturally express the syntactic structure assumed for the sentences under consideration. Treebanks may also serve as a resource for Natural Language Processing, where the goal often is to perform a certain task automatically, based on training data derived from the treebank. Since all three of annotation scheme, goal of automatic annotation, and annotation method are mutually interdependent, one or more of them may be modified to improve performance on a given task. In this paper, we focus on the task of Topological Field Parsing using a German treebank (TüBa-D/Z) as input and a Probabilistic Context-Free Grammar (PCFG) as parsing method. We deliberately choose to keep both the goal of the annotation, and the annotation method fixed, and propose to examine the effect of treebank transformations on performance. For this purpose, we apply nonterminal split and merge operations that we call *Directed Treebank Refinement* to transform the structure of a treebank, aiming at encoding the same information in a way more suitable for the parsing task at hand.

PCFGs are the backbone of many current broad coverage, high accuracy parsing systems [5, 6]. What makes PCFGs interesting for this task is that there exist efficient algorithms to find all parses given a context-free grammar (CFG), and to find the best parse given additional rule probabilities (hence a *P*CFG).

1.1 Limitations of Context-Free Natural Language Parsing

The application of a Context-Free Grammar to parsing natural language assumes that all relevant syntactic phenomena only have limited context sensitivity and that syntactic structure can be encoded by proper trees. It is well known that the latter assumption is violated for some syntactic constructions in different languages,

because some phenomena cannot be encoded without crossing edges. The former assumption depends on the kind of structure you assume for a given utterance. For example, in prepositional phrase (PP) attachment you typically have to decide whether to attach a PP to a noun phrase (NP) or to the verb dominating both PP and NP. However, normally the head words of the NP, the PP and of the including clause are not present in the labels, so that the expansion of your NP nonterminal cannot take into account the relevant context, which for PP attachment includes lexical information about the NP, PP and verbal head words [8]. In languages with less restricted word order than English, this problem tends to be still more severe, because more information relevant to disambiguate syntactic structure is present at the terminal level, but usually not represented in the node labels (e.g. morphological case markers for German). As a result, the absolute restrictions of context-free syntactic parsing may be less severe than the restrictions resulting from a certain way of encoding syntactic structure.

1.2 Adding Context-Sensitivity

Research on English using the Penn Treebank seems to converge on a set of useful additional contextual features for extending PCFGs [7, for an overview table]. However, these features seem to be language (and also annotation) dependent. For English, tree transformations incorporating the parent node improve performance for almost any kind of substructure [10]. Flattening the original structure, or introducing more levels, also has an impact on performance, which, however, is less consistently beneficial. This result agrees with another study, that analyses the impact of another set of tree transformations on PCFG performance [1]. In this study, nonterminal node labels are enriched with information concerning the parent node, the depth of embedding, and grammatical functions. Again, the most consistent improvement results from using parent nonterminal information. Another study applies an extended PCFG model to German that was originally developed for English [7]. It shows that because the annotation of their data tends to be flatter than the corresponding English annotation structure, some of the PCFG parser's extensions have to be adapted to improve performance.

All these approaches have in common that incorporating contextual information into PCFG parsing improves performance, and that the interesting context is defined *a priori*, either by conditioning the model on this context [6, 7], or by extending the node label with contextual information [10, 2]. The former approaches that integrate contextual features into their PCFG model usually define backup strategies that use a subset of information in case not all necessary information is available. The latter approaches, that extend node labels with contextual information, cannot easily apply such a backup strategy, leading to sparse data problems.

Given e.g. the parent node transform, a new node label is introduced for each combination of node label plus parent node label. This will generally result in less than $|N|^2$ new node labels for nonterminal node label set N, because annotation structure allows dominance relations only between certain sets of nonterminals. Still, a high number of new node labels is introduced, and consequently, in a supervised learning regime, the amount of manually annotated sample data per nonterminal decreases.

To summarise, there is much evidence that incorporating more contextual information into PCFGs can be beneficial. The remaining question is, which information exactly to incorporate, and how to incorporate it into PCFG parsing. We also choose to transform the structure of labelled input data to add contextual information, and we expect that the ideal transformation uses different node labels everywhere, and only where, different context causes PCFG productions to be different. We believe that Directed Treebank Refinement (DTR) presented here is able to detect this information at least partially.

2 Data and Methods

2.1 The German Treebank TüBa-D/Z

The *Tübinger Baumbank des Deutschen / Zeitung* (TüBa-D/Z) is a treebank which is manually annotated with syntactic information similar in spirit to [16], but annotating text from the newspaper *die tageszeitung* (taz) instead of spontaneous speech. It follows the Topological Field model for German [9]. Grammatical functions are annotated as edge labels (see figure 1). Crossing edges do not occur in TüBa-D/Z despite the relatively free word order of German, because the topological fields sequentially group the words of a sentence, and edge labels coindex relations that cross the fields (e.g. a relative clause modifying an accusative object receives the label `OA-MOD`, so that intervening parts of the verbal group do not disturb the proper tree). The process of annotation is not yet finished. Hence we extracted the subset of annotated sentences.[1] We also restrict all our experiments to sentences with length of up to 40 words. The TüBa-D/Z data consists of five days from the *taz* newspaper (May 3–May 7, 1999). The last two days have received most extensive checks, so that we select data from them as test data. We use a small and a large data set as shown in table 1.

[1] We assume that a sentence is annotated when each terminal has at least one dominating nonterminal.

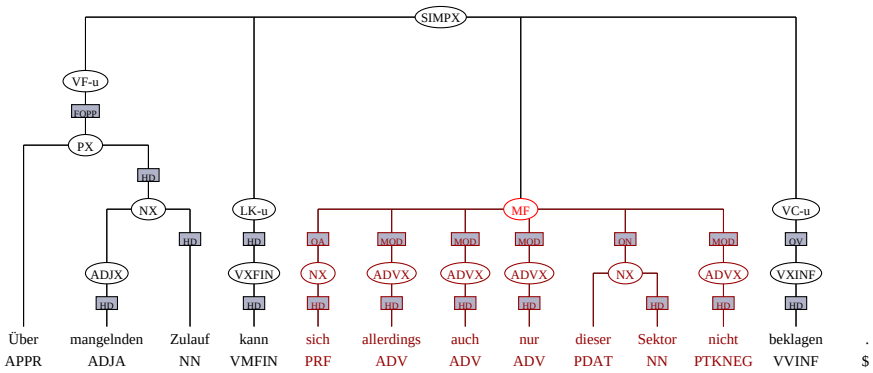

Figure 1: TüBa-D/Z sample sentence

small	sentences	words	large	sentences	words
train	7398	118650	train	10472	168996
test	3074	50346	test	4001	65834

Table 1: Sizes of large and small train/test data sets

2.2 Preprocessing

For all experiments, we drop edge labels encoding grammatical functions which therefore are effectively ignored in parsing and evaluation. Edge labels are used in TüBa-D/Z also to encode some arguably structural aspects of the annotation, including heads of constituents and the conjuncts in coordinations. We further employ edge labels in contrast to the original TüBa-D/Z scheme to represent unattached constituents, in order to obtain proper trees. We use these *structural edge labels* by adding them to the child node label in part of the experiments we perform.

There may occur *cycles*[2] in the *treebank grammar*[3] derived from TüBa-D/Z. If the cycle is present already in the treebank (as opposed to the resulting grammar), then all dominated node labels involved in the cycle are recursively added to the dominating node label (see last word in figure 2). We call this method of transforming the treebank `treeb`. Alternatively, the `gram` method attempts to find the minimal set of productions that removes cycles from the treebank *grammar*. The `gram` method generates all different cycles from the grammar and iteratively

[2]Cycles in the grammar are sets of unary productions $n_1 \rightarrow n_2, n_2 \rightarrow n_3, \ldots, n_n \rightarrow n_1$, where $n_1, \ldots n_n$ are all nonterminals.

[3]We produce a treebank grammar from a treebank by reading off CFG productions from it, and we estimate production probabilities from the relative production frequencies of nonterminals [4].

assigns the parents in them new names until no more cycles occur. Productions participating in more cycles and appearing less often in the treebank are changed first by this method.

For some experiments we *binarise* productions of nodes with two or more children into right-branching structures where the first child is left in the first level, and the remaining children are iteratively dominated by copies of the original expanded nonterminal. For handling the last remaining child we choose to introduce a last copy of the focus node and obtain a final unary production which separates the information that a node terminates from the category of the last child.

We use the PCFG implementation of `lopar`[4] for all our experiments, always using unlexicalised PCFGs, and training only in supervised mode. All experiments are performed on manually annotated POS tags.

2.3 Directed Treebank Refinement

In this section, we describe a nonterminal split heuristic introduced for a different PCFG parsing task [3] and a new nonterminal merge heuristic to the context-free productions in a treebank. The transformation is *directed* at including context into node labels wherever the distribution of the node's productions depends on this context.

2.3.1 Splitting Nonterminals

For splitting nonterminals we adopt the proposal of [3]. They use a PCFG to parse RNA sequences, modifying a handcrafted CFG by unsupervised learning using the EM algorithm on unlabelled data.[5] We use their refinement heuristic and adapt their refinement operator (calling it henceforth *split* heuristic/operator), and we add a merge heuristic that they mention but do not specify.

The split heuristic searches for nonterminals with the same label but different usage. We define the *usage* of a nonterminal (the *focus node*) to be the distribution of its *productions* when appearing in a certain *context*. In order to determine whether two nodes in context are different, we further need to define *context* and *similarity*. Following [3], we define the context of a focus node to be the label of its parent node, and we define the similarity between two nonterminals as the similarity between the distributions of their productions. In order to measure the latter, we use a standard statistical test (χ^2) and measures of cross entropy, again following [3]. We now briefly sketch computing χ^2 in

[4] `lopar` is available from `http://www.ims.uni-stuttgart.de/projekte/gramotron/SOFTWARE/LoPar-en.html`. We use `lopar` version 3.0 [15].

[5] Their data consists of positive and negative examples without structural annotation.

the split heuristic: Given a nonterminal n appearing $f(n)$ times in the corpus, we observe m different productions $p(n) = \{p_{n1}, p_{n2}, \ldots p_{nm}\}$ with frequencies $f(p_{ni})$. We calculate the probability of each production from its relative frequency $P(p_{ni}) = f(p_{ni})/f(n), i = 1, \ldots m$. When we only look at the productions of a focus node in context, we restrict ourselves to the productions of n under a certain parent node c. We can now calculate the expected frequency of production p_{ni} in context $c > n$ by multiplying the frequency of the node in context $f(n, c > n)$ with the probability of the production in the whole corpus: $f_e(p_{ni}, c > n) = P(p_{ni})f(n, c > n)$.[6] Finally χ^2_{nc} is calculated from the difference between the observed and expected values for all productions of a focus node n in context c as follows:

$$\chi^2_{nc} = \sum_{i=1}^{m} (f_e(p_{ni}, c > n) - f(p_{ni}, c > n))^2 / f_e(p_{ni}, c > n)$$

Using the χ^2-test is not possible for cases where only one production occurs consistently.[7] However, focus nodes in context that consistently have unary productions, and have different productions in other contexts, most probably should receive a new node label by the split heuristic. The Skew Divergence (SD) is an entropy based measure which is able to compare distributions even when they contain only a single production. It is a variant of the Kulback-Leibler divergence employed in [3] and more robust when not all productions are shared between the distributions [12]. Using the notation introduced above and $P^c(p_{ni})$ to denominate the probability of production p_{ni} to appear in context $c > n$, the Skew Divergence of a focus node n in context c is calculated as

$$\mathrm{SD}_{\alpha,nc} = \sum_{i=1}^{m} P^c(p_{ni})[logP^c(p_{ni}) - log(\alpha P(p_{ni}) + (1-\alpha)P^c(p_{ni}))]$$

2.3.2 Merging Nonterminals

We define a new heuristic to spot nonterminal merge candidates. As opposed to splitting nonterminals, the merge heuristic looks for nonterminals in context that are similar, i.e. two different nonterminals that have similar sets of productions when they appear below some (possibly different) parent.

When defining a merge heuristic, the requirements of the χ^2-test are very often difficult to satisfy, because both distributions representing candidates for a merge

[6] We use "$>$" to express the dominance relationship, while "A $\rightarrow$ B" denotes a production.

[7] When minimal sample frequencies are smaller than allowed for obtaining reliable predictions from the χ^2 statistical test, we merge classes and do no longer claim to obtain statistically interpretable results.

are the productions of a node in context, and sample frequencies tend to be low. We therefore only employ the SD metric for merging. As SD is asymmetric, we choose the bigger SD of both when using it to determine divergence between two merge candidates. We apply the merge operator only when similarity is transitive.[8]

Directed Treebank Refinement is a combination of the merge and the split heuristics, which can be combined in several ways, e.g. by repeatedly trying to merge nonterminals before trying to split them, until both attempts fail. However, merging nodes is more costly than splitting nodes[9], and we therefore iteratively split until χ^2/SD drops below a certain threshold, and then merge until all χ^2/SD exceed a certain threshold, both up to a maximum number of times.

3 Experiments and Results

We examine the effect of DTR on Topological Field parsing (*TopF*-Parsing), which is the task of determining the overall sentence structure which for us consists of all verbal frames, fields, and sentential nodes (see table 4).[10] As a result, all complements and adjuncts are attributed to the correct verbal group, without specifying their relation to the verb (i.e. their grammatical function) and without specifying the internal structure of these complements and adjuncts, and their relation to each other (i.e. their complex internal phrase structure). The TopF parsing task also includes coordination on the sentential and field level. Recent research seems to indicate that the topological field structure is a good starting point for parsing German, and that it can be seen as an independent parsing task because of its syntactic rather than lexical nature [14]. TopF-parsing can be considered as the successor of TopF-chunking, where the verbal parts of each sentence are detected without connecting them through sentence structure [17].

We perform three sets of experiments. All of them fully consider the attachment of punctuation and other terminals that are never attached to any constituent in the original treebank. The first experiment aims at answering the question whether DTR is capable of introducing relevant context into node labels. We binarise the treebank and apply the split and merge heuristics to it.[11] We expect that DTR returns some but not all of the original sequential relations encoded in the broken-up

[8] This means that we require that when $a \approx b$ and $b \approx c$, that also $a \approx c$, where a, b, c are nodes in context, and $\approx$ denotes similarity. These restrictions prevent infinite alternations between, e.g. merging (a, b) and (b, c).

[9] Merging involves $(|N| - 1)|N|/2$ tests for similarity, and splitting only $|N|$.

[10] `FKONJ` replaces all explicitly named sequences of fields in [16].

[11] We always set the stopping conditions for splitting to $\chi^2 > 10$ and SD > 1, iterating at most 200 times, which is reached only for split operations. Merge operations stop at a maximum of 35 iterations for SD < 0.2. We use a fixed $\alpha = 0.99$ for SD.

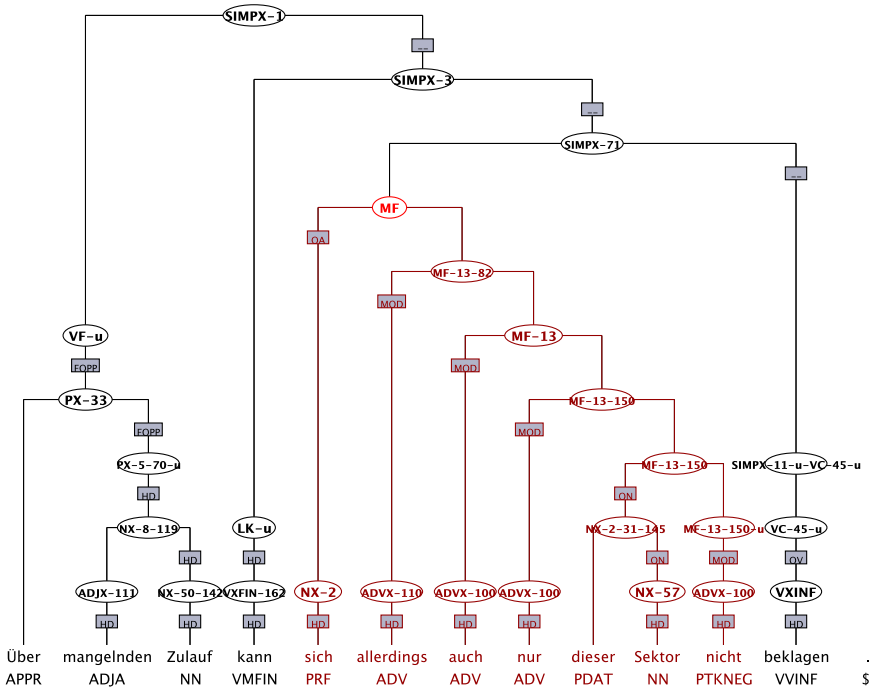

Figure 2: Binarised sample sentence after splitting nonterminals (cf. figure 1)

right hand sides of rules into the treebank. See figure 2 for the result of DTR and table 2 for the relevant split operations. The second set of experiments aims to relate differences in performance caused by differences in preprocessing, in node labels, size of training data, and application of DTR to the baseline performance of training on the unmodified TüBa-D/Z data.[12] Table 3 shows the overall results, and table 4 the performance per node label of the basline model and the best model not using additional (structural edge) input information. The last set of experiments aims at evaluating all aspects of TopF structure, including structural edge information (see table 5).

[12]The results are given as $F_{\beta=1}$ for the set of labeled and unlabeled (start position, end position) tuples of each constituent in the trees of the test set.

	nonterminal in context			most deviant productions		
it.	parent >	focus	f	$\rightarrow$ production	obs.	exp.
1	0	SIMPX	7767	VF SIMPX	6430	1649
3	SIMPX-1	SIMPX	8233	LK SIMPX	6208	2116
11	SIMPX	SIMPX	10506	VC	4051	2274
13	MF	MF	13897	NX-2 MF	1070	2572
45	SIMPX-11	VC	2597	VXFIN:43 VVPP:4 VXFIN VC:1	48	582
71	SIMPX-3	SIMPX	7760	C SIMPX-11:7 SIMPX-11 SIMPX-11:7	14	206
82	MF	MF-13	7516	PX	1494	1862
150	MF-13	MF-13	1650	ADVX-100 MF-13	158	231
162	LK	VXFIN	9081	VXFIN VXFIN:10 FM:1 VVINF:1	12	62

Table 2: Most deviant productions of split TopF nonterminals from figure 2

3.1 Discussion of Results

Looking at the individual split operations of the first experiment, the rows splitting SIMPX nodes in table 2 show the desired percolation of contextual information down the tree. First, the initial element of the main clause is split from the rest (VF), followed by those elements that naturally follow in the TopF model (LK and VC). It is also unlikely that an LK is followed by a C-field or two VCs (it. 71). We conclude that DTR is generally able to distinguish relevant contexts.

Results of the other experiments indicate that DTR may reduce the error rate considerably (overall by 30% for the experiments in table 4). DTR seems to be beneficial in cases where node labels are used consistently for a limited set of productions, e.g. in the C-field, which introduces relative clauses in German. The C-field is often occupied by relative pronouns, which are projected to noun phrases first, and only then to C-fields. Only noun phrases in the C-field ever contain relative pronouns, which is not captured well by the original distribution of noun phrase label productions. Other node types indirectly benefit from improving C field annotation, e.g. $F_{\beta=1}$ of relative clauses (R-SIMPX) increases by 45%. Only few node labels suffer from applying DTR. P-SIMPX, which has the biggest loss of 10%, only occurs 8 times in test data, however.

The performance on full structural TopF annotation (including structural edges) is only slightly worse than on the original label set (table 5). Surprisingly, the χ^2 split heuristic seems to work better in combination with the treeb de-cycling, as opposed to SD splitting that seems to combine better with the gram method. Using χ^2 splitting produces slightly better results here.

		small		large		
de-cycling		none	treeb	none	gram	treeb
original	lab.	84.98	85.33	cyclic	85.53	87.15
	unl.	88.73	87.90		89.57	89.52
edges	lab.	83.83	84.80	cyclic	85.97	88.24
	unl.	87.86	87.25		89.95	90.29
splitSD	lab.	88.19	87.36	89.86	89.86	89.57
	unl.	90.18	89.50	91.50	91.50	91.36
splitCS	lab.	87.57	86.50	cyclic	89.81	89.05
	unl.	89.73	88.92		91.59	90.93
splitSD	lab.	88.33	87.35	89.90	89.90	89.78
+ merge	unl.	90.35	89.42	91.55	91.55	91.43
edg. + splSD	lab.	88.19	84.62	90.04	90.04	88.18
	unl.	90.34	86.95	91.98	91.98	90.00
edg. + splCS	lab.	87.36	84.00	89.84	89.84	87.36
	unl.	89.62	86.60	91.65	91.66	89.46
edg. + splSD	lab.	88.22	84.63	90.12	90.12	88.26
+ merge	unl.	90.26	86.86	91.95	91.95	89.95

Table 3: Overall labelled and unlabelled $F_{\beta=1}$ for original TopF labels

label	base	best	label	base	best
overall	85.53	89.90	VC	97.83	98.41
LV	0.00	35.71	VCE	0.00	0.00
KOORD	81.24	90.80	NF	68.35	71.82
PARORD	0.00	0.00	FKOORD	48.29	47.34
VF	92.45	92.68	FKONJ	63.44	62.83
C	84.25	96.78	P-SIMPX	10.00	0.00
LK	98.81	99.26	R-SIMPX	35.12	80.91
MF	86.60	90.30	SIMPX	77.99	84.96

Table 4: $F_{\beta=1}$ per original TopF label type for best transform and baseline

	small		large		
de-cycling	none	treeb	none	gram	treeb
edges	83.15	83.97	cyclic	85.30	87.53
edges + splitSD	87.51	83.71	89.38	89.38	87.57
edges + splitCS	86.55	87.52	89.08	89.08	89.78
edges + splitSD + merge	87.53	83.72	89.49	89.49	87.64

Table 5: Overall labelled $F_{\beta=1}$ for TopF labels with structural edges

4 Conclusion and Future Research

We have applied a modified nonterminal split heuristic as presented in [3] and a new merge heuristic to a PCFG parsing task that assigns complex clause structure to POS tagged input, when trained on a hand-labelled syntactic treebank of German. Qualitative evaluation on binarised data suggests that DTR is able to include relevant contextual information into nonterminal node labels, and quantitative evaluation shows a performance gain when using DTR before generating the treebank grammar for training the PCFG.

In order to compare results obtained on different grammar structures, we would like to apply an evaluation metric that normalises for different structures. Dependency based evaluation lends itself naturally to this task [13, 11], and we therefore plan to apply it. DTR can be applied to any corpus for which focus nodes, contexts, and productions can be defined. It will, however, be most useful to support parsing strategies with limited context sensitivity like PCFGs, or as a means to detect unexpected usages of nodes, which in the development phase of a treebank often turn out to be errors. The latter may be easily performed also on corpora using graphs instead of trees, which is another direction of future research.

Acknowledgements

This research was supported by the German Research Council (DFG) as part of *Sonderforschungsbereich 441: Linguistische Datenstrukturen*. I would like to thank Kiril Simov for letting me visit his department at the Bulgarian Academy of Sciences where the basic idea for this paper evolved. Many thanks also to Jorn Veenstra for valuable comments and discussions on earlier versions of the paper.

References

[1] Anja Belz. Optimisation of corpus-derived probabilistic grammars. In *Proceedings of Corpus Linguistics*, pages 46–57, 2001.

[2] Anja Belz. PCFG learning by nonterminal partition search. In *Proceedings of ICGI 2002*, pages 14–27, Berlin, 2002. Springer.

[3] Joseph Bockhorst and Mark Craven. Refining the structure of a stochastic context-free grammar. In *Proceedings of IJCAI-2001*, 2001.

[4] Eugene Charniak. Tree-bank grammars. Technical Report CS-96-02, Brown University, Department of Computer Science, 1996.

[5] Eugene Charniak. A maximum-entropy-inspired parser. Technical Report CS-99-12, Brown University, 1999.

[6] Michael Collins. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, 1999.

[7] Amit Dubey and Frank Keller. Probabilistic parsing for German using sisterhead dependencies. In *Proceedings of ACL*, 2003.

[8] Donald Hindle and Mats Rooth. Structural ambiguity and lexical relations. *Computational Linguistics*, 19(1):103–120, 1993.

[9] Tilman Höhle. Der Begriff 'Mittelfeld', Anmerkungen über die Theorie der topologischen Felder. In *Akten des Siebten Internationalen Germanistenkongresses 1985*, pages 329–340, Göttingen, 1986.

[10] Mark Johnson. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632, 1998.

[11] Sandra Kübler and Heike Telljohann. Towards a dependency-oriented evaluation for partial parsing. In *Proceedings of Beyond PARSEVAL (LREC Workshop)*, Las Palmas, Gran Canaria, June 2002.

[12] Lillian Lee. On the effectiveness of the skew divergence for statistical language analysis. In *Artificial Intelligence and Statistics 2001*, pages 65–72, 2001.

[13] Dekang Lin. A dependency-based method for evaluating broad-coverage parsers. *Natural Language Engineering*, 4(2):97–114, 1998.

[14] Frank Henrik Müller and Tylman Ule. Annotating topological fields and chunks – and revising POS tags at the same time. In *Proceedings of COLING-2002*, 2002.

[15] Helmut Schmid. Lopar: Design and implementation. Technical report, IMS, Universität Stuttgart, 2000. Arbeitspapiere des Sonderforschungsbereichs 340.

[16] Rosmary Stegmann, Heike Telljohann, and Erhard W. Hinrichs. Stylebook for the German treebank in VERBMOBIL. Verbmobil-Report 239, Eberhard-Karls-Universität Tübingen, September 2000.

[17] Jorn Veenstra, Frank Henrik Müller, and Tylman Ule. Topological Fields Chunking for German. In *Proceedings of CoNLL-2002*, 2002.