In: Proceedings of the 14th Meeting of Computational Linguistics in the Netherlands, Antwerp, Belgium, 2003.

Iterative Treebank Refinement

Tylman Ule and Jorn Veenstra

Seminar für Sprachwissenschaft, Universität Tübingen

Abstract

Treebanks are a valuable resource for the training of parsers that perform automatic annotation of unseen data. It has been shown that changes in the representation of linguistic annotation have an impact on the performance of a certain annotation task. We focus on the task of Topological Field Parsing for German using Probabilistic Context-Free Grammars in the present research. We investigate an iterative algorithm for tuning the label set of a given treebank to this task and show that the number of parses proposed by a context-free grammar is reduced considerably in addition to an increase in labelled precision and recall for the annotation of node labels. We also show that the optimal refinement can be achieved with a relatively small number of changes to the treebank.

1 Introduction

Treebanks have undisputed value for many purposes, one of them being to train parsers. That being only one of all possible applications, the design of a treebank has to fulfil other needs as well. The main purpose of a treebank is to reflect the assumptions of their designers about what linguistic information is made explicit by the annotation. However, an important design criterion is also that the underlying annotation scheme is stated clearly in guidelines and is easily accessible, so that annotators can follow these guidelines to produce a corpus that is consistently annotated with high accuracy. The choice of tools available also influences the representation of linguistic annotation. It is unlikely that the design of a treebank meets all these requirements equally well, e.g. that a corpus whose annotation is optimised for representing linguistic information will be optimally tailored for training parsers.

In this paper, we focus on the relation between a specific application and the shape of linguistic annotation in a corpus. We annotate the complex structure of topological fields in German using context-free grammars. Previous experiments have shown that the label set does not optimally reflect distinctions between distributions of productions, and that few modifications may have considerable impact on parsing accuracy (Veenstra, Müller and Ule 2002). Our goal here is twofold. First, we try to find the minimal set of distribution-driven treebank transformations that yield optimal parsing performance. Second, we examine whether the transformations can be used as an efficient means to reduce the number of parses generated by the context-free grammar.

2 Data and Methods

2.1 A Corpus of Topological Fields for German

The Topological Fields (TopF) Model for German offers an approach to handle constituent order in German, which is relatively free (Höhle 1986). Complex German sentences are analysed according to the model as a hierarchy of verbal and non-verbal fields. The finite and non-finite parts of the verb make up the sentence bracket (LK and VC in the subclause of figure 1). The TopF model groups complements and adjuncts into the same clause and puts them into the front (VF), middle (MF), or final field (NF). A sentence fully annotated with topological fields limits the search space when determining e.g. grammatical relations between the phrases and the verbal parts of a sentence. In figure 1, e.g., the verb in the matrix clause may only have relations to either the sentential object (marked by the edge label OS) or the pronoun *sie* (ON; nominative object). In the subordinate clause, only strings or substrings of *Ercettin* and *Pop mit Niveau* may have a grammatical function. Figure 1 shows all grammatical function edge labels in light grey, and all TopF labels have heavy outlines. The TopF model seems to be descriptively adequate (Eisenberg 1999) and it can also be implemented efficiently (Becker and Frank 2002, Veenstra et al. 2002).

We base our experiments on a corpus which has been annotated with topological fields: The Tübinger Baumbank des Deutschen / Zeitung (TüBa-D/Z) (Telljohann, Hinrichs and Kübler 2003).[1] It consists of manually annotated texts from the newspaper *taz*. In addition to topological fields, the annotation also includes constituent structure, where the maximal projections are marked with their grammatical function in the clause. We use a tune and a final data set for our experiments, which both are divided into test and training data as shown in table 1.[2]

Tune	Sentences	Words
Train	7624	122661
Test	3067	50327

Final	Sentences	Words
Train	10691	172988
Test	4009	65868

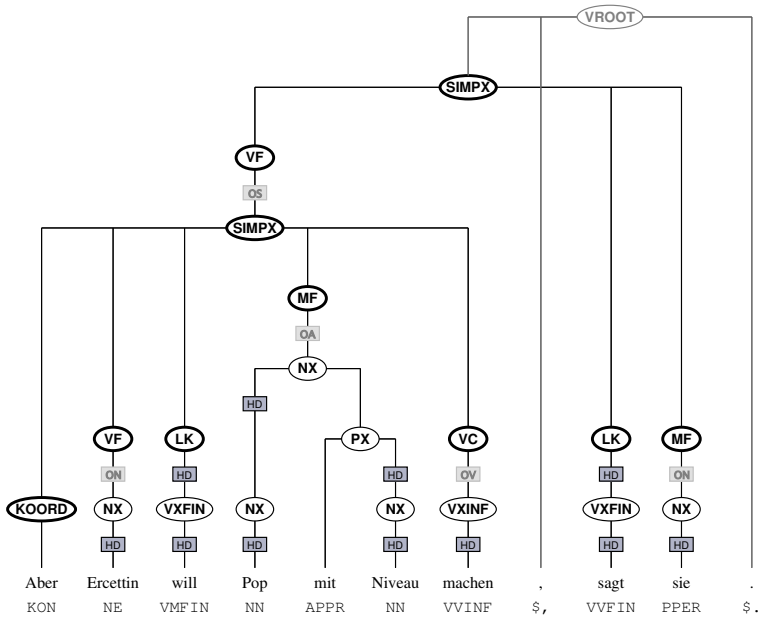
Table 1: Sizes of Tune and Final Data Sets

TüBa-D/Z uses the *export* data model to describe the annotation of sentences (Brants 1997). In the export model, linguistic annotation is represented by directed acyclic graphs with labelled nodes and edges.[3] Annotation in TüBa-D/Z is restricted to non-crossing edges, so that the annotation graphs form proper trees. In the present context, we evaluate the adequacy of an annotation scheme for annotating topological fields using context-free grammars (CFGs). CFGs only dis-

[1] TüBa-D/Z is available at `http://www.sfs.uni-tuebingen.de/de_tuebadz.shtml`.

[2] The division of the data sets follows the division of TüBa-D/Z into days – May 7 is the tune test data, and May 3–5 is tune train data. Final test data is May 6, and final train data is May 3–5 and 7.

[3] The export data model also allows arbitrary directed arcs between any two nodes in a tree. In TüBa-D/Z, these *secondary edges* are used infrequently, and we ignore them here.



"But Ercettin wants pop_music with class to_do, says she."
Ercettin, however, wants to do classy pop music, she says.

Figure 1: Topological Fields in a Complex Sentence

tinguish between nodes and their daughters, i.e. edges are not labelled explicitly. As a consequence, any edge labels to be considered in the context-free representation of TüBa-D/Z have to be added to the label of the daughter node from which the labelled edge starts. Therefore we perform a number of initial transformations which we describe next.

Heads in complex phrases are explicitly marked by the edge label HD in TüBa-D/Z. When phrases or fields are coordinated, they receive the edge label KONJ. Phrases that belong to an appositional structure receive the edge label APP. We keep these labels, because they seem to be informative about TopF-structure (Ule 2003). When they occur above pre-terminals (i.e. POS tags), then a new node with the label POS+EDGE is added above the POS tag, so that this information can be added to given POS tags during parsing. In the above example, e.g., *Ercettin*/NE will be dominated by an additional NE+HD. Phrases inside fields are annotated with their grammatical functions in TüBa-D/Z (ON, OA and OS in figure 1). We do not use any information related to grammatical functions (i.e. all light grey edge labels in figure 1). Generally *using* edge labels means appending them to the node labels they dominate, while *ignoring* them means using node labels as is.

Some terminal and non-terminal nodes are not attached to any parent node, which in the export model corresponds to being attached to the special VROOT node. VROOT is the unique topmost node in all sentences. In figure 1, e.g., three nodes are not attached to a parent node: the SIMPX root node of the syntactic annotation and the two punctuation marks (comma and full stop). A context-free representation only has a single start symbol, so that the unattached terminals have to be attached to that start symbol as well. We use VROOT as the CFG start symbol and reattach all nodes (terminal or non-terminal) that are not explicitly attached to a parent node to the lowest node that dominates both its right and left sister. I.e. the comma is attached to the matrix clause (SIMPX) between VF and LK. If there is only one sister (see the full stop in figure 1) then the node is only dominated by VROOT, to which it is attached. There are also unattached non-terminal nodes in TüBa-D/Z (mostly parentheses) which are transformed into proper trees along the same lines.

Proper names are annotated in TüBa-D/Z by a single new node dominating the lowest single node in the syntactic annotation tree that spans all terminals of the proper name. If there is no node in the syntactic tree that dominates all and only the nodes that belong to a single named entity, then the full extent of the named entity is marked by secondary edges, which we ignore (for more details, see Telljohann et al. 2003).

2.2 Parser and POS

We use the `lopar` parser for all our experiments, and unlexicalised rule frequencies read directly off the treebank (Charniak 1996).[4] We use automatically part-of-speech (POS) tagged data for our experiments by training the `tnt` tagger (Brants 2000) on manually annotated texts.[5] We perform POS tagging as a separate task prior to TopF-parsing, i.e. the input to all experiments consists of sequences of automatically annotated POS tags. We use the STTS German tagset, which is also used in TüBa-D/Z (Schiller, Teufel and Thielen 1995).

2.3 Treebank Refinement

Our goal is tuning a corpus to a task, an approach that we term Treebank Refinement (TR). In the present context, the corpus is TüBa-D/Z, and the task is TopF-parsing using PCFGs. An inherent property of CFGs is context-freeness, which means that the only way to pass information about daughters to grandparents is the parent node. Figure 2 schematically shows a node label XP that has a number of different productions (i.e. sequences of daughter nodes) $\{a, \ldots, h, \ldots\}$ in the treebank. These productions have a certain distribution of occurrences, e.g. over the whole corpus b and c may be more frequent than g and h, which is schematically

[4] `lopar` is available at `http://www.ims.uni-stuttgart.de/projekte/gramotron/SOFTWARE/LoPar-en.html` (Schmid 2000).

[5] We annotate the treebank by splitting the data into ten parts, annotating each part with a model trained on the remaining nine parts. Accuracy was best when supplementing this training data with more data from the *taz* newspaper and other sources (Ule and Müller 2004).

shown by the shape of the curve above a ...h on the left-hand side of the figure. In a PCFG, every time the node label XP is expanded, this will be the underlying distribution. There may be a wider context Y in which XP occurs, however, that limits the set of productions so that the distribution of these productions shows a different shape (right half of figure 2).

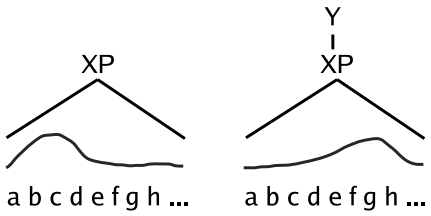


Figure 2: Node Label XP is misleading in Context-Free Grammar

TR aims at spotting productions of nonterminal nodes in treebanks that do not behave as expected when they appear in certain contexts. TR looks at all types of nonterminal nodes f in a treebank (which we call the *focus* nodes) and compares the distribution of each of f's productions over the whole treebank with its productions when appearing under a certain parent node (the *context* type c of *focus* type f). TR is applied iteratively, and in each iteration it delivers a single f that has the most unexpected distribution of productions in a certain context c.

We compare the distributions of the production types p of node f, where production means sequence of direct children. In order to compare these distributions, we employ the *Skew Divergence* (Lee 2001), which is a smoothed version of the information-based Kulback-Leibler divergence.[6]

We first introduce some notation:

$f \in \mathcal{F}$ is a *focus* type, where $\mathcal{F}$ is the set of different nonterminal node types in the treebank

$c \in \mathcal{C}_f$ is a context type of focus f, where $\mathcal{C}_f$ is the set of different context types of focus f; we consider parent nodes (including VROOT) as context so that c dominates f (in short: $c > f$)

$p \in \mathcal{P}_f$ is a production type of focus f, where $\mathcal{P}_f$ is the set of different types of productions of focus f, and a production is the sequence of direct children (in short: $f \rightarrow p$)

$prob(c > f \rightarrow p)$ is the probability that f has production p in context c, estimated via maximum likelihood

[6]Using Skew Divergence instead of the unsmoothed Kullback-Leibler Divergence would strictly be necessary only when comparing two distributions in context for merging previously split nodes, where the support of one distribution is not always a proper subset of the support of the other distribution. We nonetheless always employ Skew Divergence in order to keep all results comparable.

$prob(f \rightarrow p)$ is the probability that f has production p, estimated via maximum likelihood

α is a smoothing factor

We can now describe the divergence SD_{α}^{fc} between the distribution of productions p of node type f over the whole corpus and the distribution of its productions when f occurs in a certain context type c as:

$$SD_{\alpha}^{fc} = \sum_{p \in \mathcal{P}_f} prob(c > f \rightarrow p) \, log \frac{prob(c > f \rightarrow p)}{\alpha \, prob(f \rightarrow p) + (1 - \alpha) \, prob(c > f \rightarrow p)}$$

TR is an iterative process that determines one focus node in context that shows the most deviant distribution per iteration and assigns all occurrences of this focus node a new name, effectively splitting it. A first goal could be to find $\underset{fc}{argmax} \, SD_{\alpha}^{fc}$ in each iteration.[7] In contrast to Bockhorst and Craven (2001), who have introduced the idea to detect overloaded node labels by comparing the distribution of productions in certain contexts in a PCFG parsing task that relies on unsupervised training, we have an annotated treebank at hand. We assume that the order in which focus nodes are split should not only be influenced by the difference in distributions described by SD_{α}^{fc}, but also by the number of times a focus node occurs in a corpus, because nodes appearing more frequently will have bigger impact on performance than those occurring less often. We therefore weigh SD_{α}^{fc} by $obsfreq(c > f)$, i.e. by the number of times f is observed in context c. We also use a threshold of $obsfreq(c > f) > 10$ and choose to emphasise the impact of large Skew Divergence by using a factor of $2^{SD_{\alpha}^{fc}}$. For each iteration, we select $\hat{f}$ and $\hat{c}$ according to

$$\underset{fc}{argmax} \, (obsfreq(c > f) - 10)(2^{SD_{\alpha}^{fc}} - 1)$$

The resulting weight $\hat{w}$ denotes the maximum frequency-weighted divergence for the current iteration, which is observed for focus node type $\hat{f}$ in context type $\hat{c}$.[8] At the end of each iteration, all $\hat{f}$ in context $\hat{c}$ receive the same new label type. Remapping the new labels to the previous labels makes parses of the refined treebank directly comparable to original parses. Pseudo-code for TR is given in algorithm 1.

3 Experiments and Results

It has been shown that TR is capable of tuning a treebank for PCFG parsing when evaluated by labelled precision and recall (Ule 2003). We try to address two new

[7] This amounts to the selection metric used in Ule (2003).

[8] Subtracting 10 from $obsfreq(c > f)$ and 1 from $2^{SD_{\alpha}^{fc}}$ does not change the order of all $\hat{w}$, but it allows for skipping easily all $\hat{w} \leq 0$ which mark infrequent or non-deviant distributions.

Algorithm 1 Treebank Refinement

```
repeat
    ŵ ← 0
    for all f ∈ F do
        for all c ∈ C_f do
            d ← SD_α^fc
            w ← (obsfreq(c > f) − 10)(2^d − 1)
            if w > ŵ then
                (ŵ, f̂, ĉ) ← (w, f, c)
            end if
        end for
    end for
    if ŵ ≥ threshold then
        RENAMEFOCUSINCONTEXT(f̂, ĉ)
    end if
until ŵ < threshold
```

questions. First, which impact does each iteration of TR have on parsing accuracy, and second, does adding context into node labels mean that the ambiguity of a context-free treebank grammar is reduced?

Following Ule (2003), we define TopF-parsing as the task of assigning the set of labelled nodes given in table 2 to a string of POS. POS tagging is performed automatically prior to parsing at an accuracy of 96.36%. For all experiments, we set $\alpha = 0.99$, because results reported in Lee (2001) suggest that a high value of α performs well across a range of experimental settings.

We apply increasing numbers of iterations of TR to the training data, extract a treebank grammar from the training data, and use this grammar and `lopar` to parse the test set. The node labels of the test set are remapped to the original node labels for evaluation. First, we analyse performance for every single iteration stopping when $\hat{w} < 150$. We choose this threshold because it results in more than 100 iterations, and preliminary experiments have shown major changes in the accuracy before iteration 50. We also perform two more experiments for $\hat{w} \geq 10$ and $\hat{w} \geq 0.01$, but for the last two experiments we only train the PCFG once on the train set which has been transformed by the corresponding maximum iterations of TR. We restrict all our experiments to sentences with up to 40 words.

3.1 Iterations of TR

The graphs in figure 3 show $F_{\beta=1}$ after each iteration of TR for each node type in the set of TopF nodes except for PARORD, P-SIMPX and VCE, which never score above zero. The performance graphs can be roughly subdivided into three groups, according to the shapes of the curves. First, there are node label types that show a sharp increase in $F_{\beta=1}$ during TR. This group is shown separately in the upper graph. The upper half of the lower graph shows the label types with

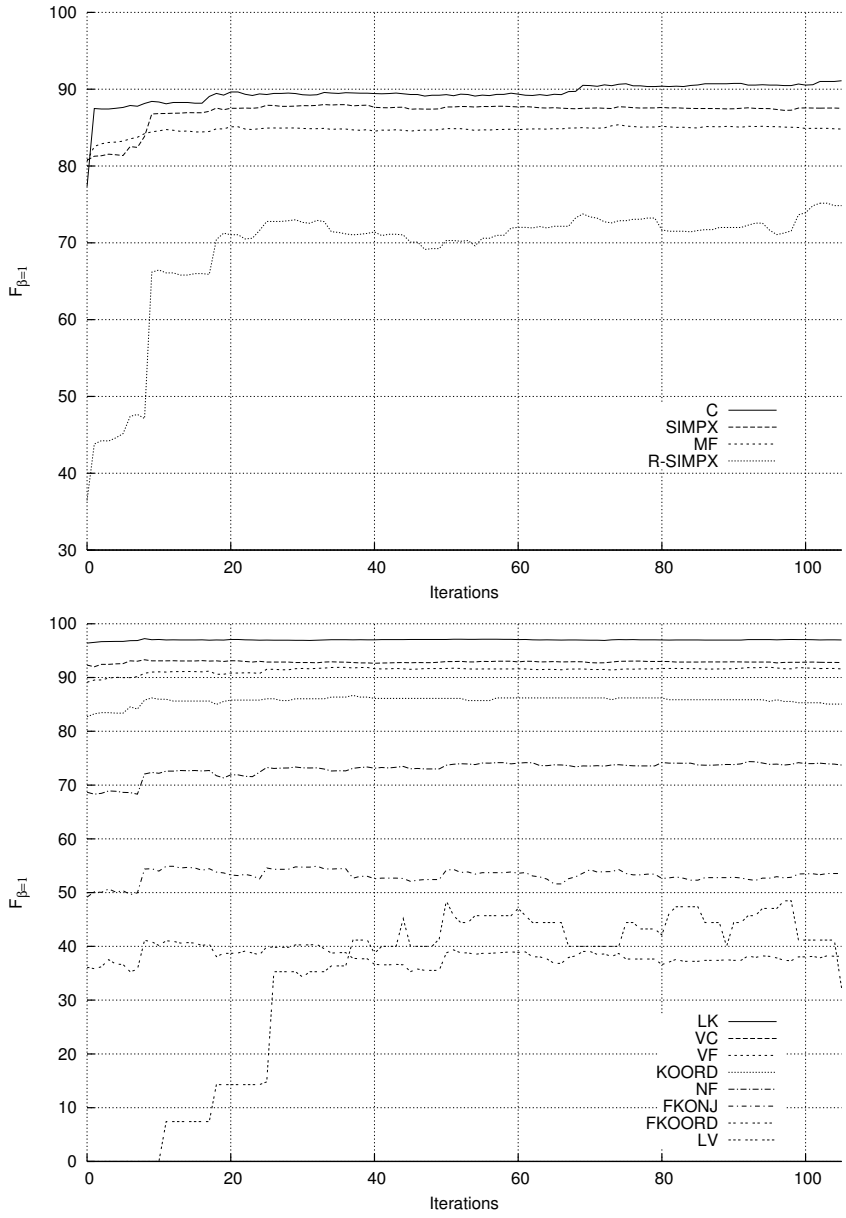


Figure 3: $F_{\beta=1}$ for TopF Node Labels by Iterations of Treebank Refinement

relatively constant performance (though never worse than baseline performance). There are also some node label types in the lower half of the lower graph that do not perform well. Table 2 shows that the four worst performing label types are also least frequent in the corpus, indicating that more data would be helpful here.[9] The two remaining types of nodes that perform far below average (FKONJ and FKOORD) are involved in coordinations, representing rather hard problems. Many of the curves increase over many iterations of TR (e.g. R-SIMPX from it. 1 to 18), which indicates that a number of transformations work together to improve labelled precision and recall.

Label	Gold	Max.	It.
all	21071	18524	74
MF	4486	3834	74
SIMPX	4330	3811	33
LK	3291	3181	8
VF	3065	2814	95
VC	2727	2555	8
NF	997	752	92
C	992	896	105

Label	Gold	Max.	It.
FKONJ	398	240	32
R-SIMPX	353	257	102
FKOORD	207	95	8
KOORD	168	153	37
PARORD	24	0	0
LV	20	8	50
P-SIMPX	13	0	0
VCE	0	0	0

Table 2: Correct Labelled Nodes in Gold Data for Optimal Tune-Test Iteration

There are noticeable improvements in figure 3 where certain node labels profit from a single iteration. Table 3 shows some of these iterations, and the biggest single contributor to SD_{α}^{fc} (column *unexp. p*).[10]

It.	$F_{\beta=1}$ increases for	$c > f$	*unexp. p*	exp	obs
1	C, R-SIMPX	C > NX	PRELS+HD	19.05	575
8	NF, FKONJ, FKOORD	NF > SIMPX	C MF VC	184.14	697
9	SIMPX, R-SIMPX	R-SIMPX > C	NX#1	253.18	625
18	R-SIMPX	SIMPX#8 > VC	VXINF+HD	154.55	415
26	LV	LV > SIMPX	C MF VC	0.79	21

Table 3: Iterations with Large $\Delta F_{\beta=1}$ for certain Node Labels

Figure 4 relates overall $F_{\beta=1}$ and $\hat{w}$. Optimal performance on the tune data set is achieved after iteration 74, which corresponds to $\hat{w} \geq 260.02$. We accordingly use this $\hat{w}$ as the stopping condition on the final data set, where $\hat{w}$ falls below this threshold after iteration 100. Overall $F_{\beta=1}$ converges rather soon to almost optimal performance and stays at a high level over many iterations.

[9] The Max. column corresponds to the best performance according to $F_{\beta=1}$.

[10] The exp column shows $expfreq(c > f) = obsfreq(f \rightarrow p)\frac{obsfreq(c>f)}{obsfreq(f)}$, i.e. the expected number of times a production of a focus node appears in this context. The obs column shows $obsfreq(c > f)$. Nodes with trailing #n have been renamed in iteration n.

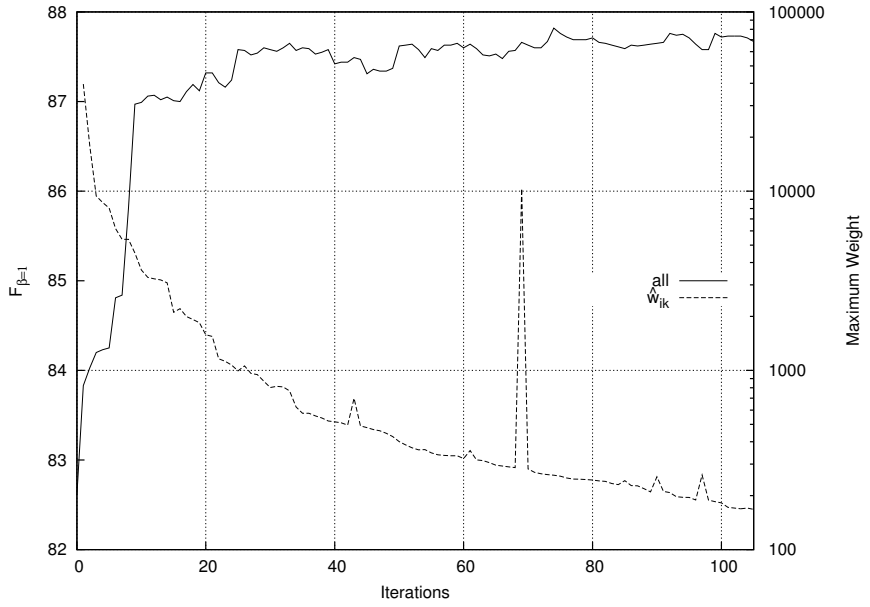


Figure 4: Overall TopF $F_{\beta=1}$ and Maximum Weight $\hat{w}$ by Iteration

3.2 Distribution of the Number of Parses

In a second evaluation, we take the decrease in the number of analyses for a sentence to measure the quality of the TR transformations, because we assume that it is related to the CFG's ambiguity. Figure 5 shows the distributions of the numbers of parses for different numbers of iterations of TR on the tune test data as box-and-whiskers plots. It includes every 10th iteration up to the stopping condition of $\hat{w} \geq 150$ (left-hand side). It also shows iteration 74, which performs best on the tune data set, and the distributions of the final iterations for $\hat{w} \geq 150$, $\hat{w} \geq 10$ and $\hat{w} \geq 0.01$. Performance of parent encoding according to Johnson (1998) is also given (JP).[11]

We summarise the previous figures in tables 4 and 5 for characteristic numbers of TR iterations for both the tune and full data sets. We include grammar sizes (# Rules) and coverage (in terms of the number of unparsed sentences; column Unparsed) for these trials. The maximum number of analyses row (Max.) shows outliers, which can also be seen in figure 5, where the whiskers are very short in upward direction because they are shown against a logarithmic scale. Everything above the whiskers denotes outliers of the distribution of numbers of analyses per parse in the test set. Their high number can be explained by the relation between sentence length and number of analyses, which is exponential for CFGs. We also show the median of the distribution of numbers of parses in the test set (Med.), the

[11] The JP transform adds the parent node label to each child over all nodes of the treebank.

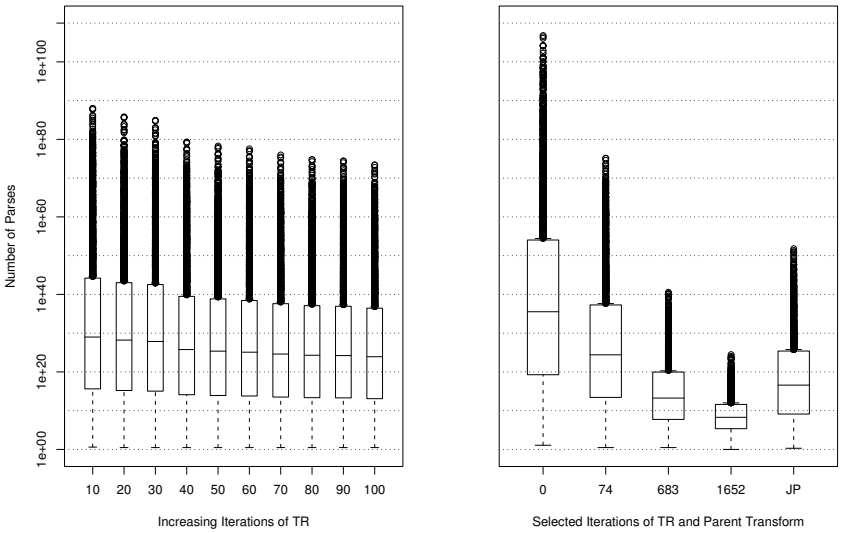


Figure 5: Numbers of Parses by Iterations of Treebank Refinement on Tune Data Set

iteration after which the trial ends (Iteration), and the labelled $F_{\beta=1}$ on the test set (lab. $F_{\beta=1}$). The w-best column of table 5 shows the result when $\hat{w}$ determined on the tune set is used as stopping condition on the final set. The last columns show results for the parent transform (JP).

Trial	base	best	150	10	0.01	JP
Iteration	0	74	105	683	1652	n/a
lab. $F_{\beta=1}$	82.6	87.8	87.7	84.5	82.8	87.5
Med. log_{10}	35.53	24.41	23.81	13.22	8.23	16.56
Max. log_{10}	106.79	75.14	73.25	40.56	24.47	51.80
Unparsed	37	39	40	47	61	44
# Rules	3943	6438	7221	13316	16678	7403

Table 4: Evaluation on Tune Data Set

The median of the distributions seems to decrease monotonically over increasing numbers of iterations of TR. The experiments for much higher numbers of iterations support this observation, yielding a median of less than 10^{10} parses compared to more than 10^{30} parses of the baseline model. The highest number of parses drops accordingly. The JP transform also shows a noticeable drop in median and maximum number of parses. Higher numbers of iterations tend to

Trial	base	w-best	150	10	0.01	JP
Iteration	0	100	136	895	2101	n/a
lab. $F_{\beta=1}$	80.5	89.2	89.2	86.7	85.6	88.8
Med. log_{10}	33.96	26.76	26.64	13.45	9.92	18.92
Max. log_{10}	109.47	87.15	84.95	41.49	29.50	62.11
Unparsed	84	86	86	90	103	90
# Rules	4814	8400	9259	17378	21624	9040

Table 5: Evaluation on Full Data Set

reduce the number of analyses for all sentences, where for the first few iterations including the best performing iteration, a major reduction can already be observed. Stopping at w-test seems to be a viable approach, because it achieves the highest performance at a smaller number of iterations than $\hat{w} \geq 150$. Performance of all transformations is better than the baseline models. The number of unparsed sentences on the final data set is much higher than expected from the difference in size between the tune test and final test sets, suggesting differences in quality between the data sets. The poor performance of the final base model may have to do with the high number of rules.

The experiments suggest that it is useful to investigate the iterative behaviour of TR because it seems to be capable of increasing labelled precision and recall on the set of annotated node labels quickly after few iterations, and also to reduce considerably the number of parses proposed by a CFG read off the resulting treebank. For all TR experiments we perform, $F_{\beta=1}$ turns out to be superior to that of the baseline model, and for the highest number of iterations we perform, the median of the distribution of the numbers of parses for the tune set is reduced from 10^{35} to 10^{8}, and quite similarly from 10^{34} to 10^{10} for the full data set. Performance of the JP transform is slightly worse than for the optimal iteration of TR with respect to $F_{\beta=1}$, coverage, and grammar size. However, the number of parses is lower for TR only after a high number of iterations.

3.3 Related Research

Johnson (1998) shows that simple transformations in a treebank can considerably improve parsing accuracy of CFGs. He adds context to all nodes by adding parent labels to node labels everywhere and at once. In TR, this would be an unconditional split of all nodes in a single iteration. The simple parent encoding performs surprisingly well in comparison to TR. We expect the difference to widen, however, when extending the notion of context beyond parent nodes to, e.g., parents and grandparents.

Klein and Manning (2003) push unlexicalised PCFG performance beyond early lexicalised PCFGs by selectively relabelling nodes. They include internal (head word) and external (ancestor) context into node labels, stressing the importance

of passing information across several levels of CFG productions. They do not only split non-terminal node labels, but also pre-terminals (i.e. POS tags), and closed lexical classes. Their changes are linguistically motivated, and applied by manual hill-climb. While TR as reported here cannot detect all of the deviations in distributions that Klein and Manning exploit, we believe that all distinctions can be incorporated by extending the notion of context, focus node and productions straightforwardly.

Belz (2002) presents a merge operator and applies beam search to derive an optimal partitioning of an initial grammar according to an objective function. The partitioning is shown to include useful parent information into node labels. In the experiment reported, partitioning can produce the same grammar as Treebank Refinement as long as the context triggering a split in TR is not the result of a previous iteration. Partition search can model the interactions between iterations of TR by extending the initial grammar accordingly, which becomes more costly when more context is considered in TR.

While all the above experiments have been performed on English data from the Penn Treebank, Becker and Frank (2002) use PCFGs to annotate topological fields in German. They use the *negra* corpus (Skut, Krenn, Brants and Uszkoreit 1997) as training material. TopF structure is not natively encoded in their corpus, so that an automatic conversion to TopFs is first applied to it, which yields 93.0%/93.7% labelled precision/recall. This data is used to generate a treebank grammar for training a PCFG and for testing the resulting grammar. They vary the information retained in the node labels, the handling of punctuation, and binarisation and pruning of the grammar, all of which they apply deterministically. While text type and label set of Becker and Frank (2002) are similar to what is used in the present experiments, TüBa-D/Z has been annotated manually. The data of Becker and Frank (2002) is produced automatically with an accuracy resembling the performance of their annotation method, which makes it difficult to compare with results obtained on manually annotated data (they obtain a maximum 93.4%/92.9% labelled precision/recall using gold POS by applying all proposed transformations).

3.4 Future Research

We have seen that TR reduces the number of parses that a naive CFG assigns to a POS sequence. Additionally, the labelled bracketings of nodes seem to be recoverable from transformed data at least as easily as they are from untransformed data using a plain PCFG. TR with an extended notion of context may be able to recover larger fragments of relevant data given the same amount of data. We expect that extended context will also allow to apply TR to improve more detailed annotation of treebanks, including full phrasal annotation, and grammatical functions. We also expect that the difference in performance relative to the JP transform will be more noticeable.

We therefore reckon that it may well be possible to include enough relevant context into node labels of a treebank, so that an otherwise context-free method will be able to enumerate a reduced set of parses that is small enough to serve as

input to much more powerful disambiguation methods (as proposed by e.g. Collins and Duffy 2002). In order to see whether the reduced set of parses is useful as preprocessing, it will be necessary to show that the gold parses are still in the reduced set of parses resulting from TR, which we plan to investigate further.

With increasing numbers of iterations, the sparse data problem becomes more pressing because the number of node labels, and therefore the number of rules in the grammar increases. We expect that more data will help us overcome this problem, but rather than proposing to annotate more data we think that in this context punctuation plays an important role. Punctuation marks are currently used as encoded in TüBa-D/Z, where no information is given as to where a certain punctuation mark belongs. Preliminary experiments with simple heuristics that attach e.g. commas to a following relative clause have shown a positive impact on performance. We expect that handling punctuation will reduce the number of rules, because punctuation can be attached much lower in the tree, making the resulting rules more regular.

We expect that refinement exclusively relying on splitting node labels will unnecessarily intensify sparse data problems. We therefore plan to develop a merge operator that does not further subdivide distributions as does the merge operator presented in Ule (2003). Merging there looks for any two node labels in any two contexts that have the same set of productions and assigns these two node labels the same new name. We expect that an improved *merge* operator will be advantageous, so that distributions of productions in different areas of the sentence that have been split independently will be merged again when equal, so that again, the number of examples per class increases.

4 Conclusion

We have shown that treebank transformations that aim at encoding differences in expansion distributions depending on a wider context in the node labels have desirable effects on treebank grammars. They can improve parsing accuracy of a PCFG on the one hand, and reduce the number of analyses proposed by the underlying CFG on the other. While accuracy and ambiguity are not independent, a reduction in ambiguity by 10^{20} can be achieved without harming accuracy. Comparison with related research suggests that it will be useful to extend TR to consider wider context for inclusion into node labels.

Acknowledgements

This research was supported by the German Research Council (DFG) as part of the research program *Sonderforschungsbereich 441: Linguistische Datenstrukturen*. We would like to thank the anonymous reviewers for their helpful comments.

References

Becker, M. and Frank, A.(2002), A stochastic topological parser for German, *Proceedings of COLING-2002*.

Belz, A.(2002), PCFG learning by nonterminal partition search, *Proceedings of ICGI-2002*, Springer, Berlin, pp. 14–27.

Bockhorst, J. and Craven, M.(2001), Refining the structure of a stochastic context-free grammar, *Proceedings of IJCAI-2001*.

Brants, T.(1997), The NeGra export format for annotated corpora, *Technical report*, Universität des Saarlandes. Computerlinguistik.

Brants, T.(2000), TnT – a statistical part-of-speech tagger, *Proceedings of ANLP-2000*, Seattle, WA.

Charniak, E.(1996), Tree-bank grammars, *Technical Report CS-96-02*, Brown University, Department of Computer Science.

Collins, M. and Duffy, N.(2002), Convolution kernels for natural language, *in* T. G. Dietterich, S. Becker and Z. Ghahramani (eds), *Advances in Neural Information Processing Systems 14*, MIT Press, Cambridge, MA.

Eisenberg, P.(1999), *Grundriß der deutschen Grammatik*, Vol. 2: Der Satz, Metzler, Stuttgart.

Höhle, T.(1986), Der Begriff 'Mittelfeld', Anmerkungen über die Theorie der topologischen Felder, *Akten des Siebten Internationalen Germanistenkongresses 1985*, Göttingen, pp. 329–340.

Johnson, M.(1998), PCFG models of linguistic tree representations, *Computational Linguistics* **24**(4), 613–632.

Klein, D. and Manning, C.(2003), Accurate unlexicalized parsing, *Proceedings of ACL-2003*, Sapporo, Japan, pp. 423–430.

Lee, L.(2001), On the effectiveness of the skew divergence for statistical language analysis, *Proceedings of Artificial Intelligence and Statistics 2001*, pp. 65–72.

Schiller, A., Teufel, S. and Thielen, C.(1995), Guidelines für das Taggen deutscher Textcorpora mit STTS, *Draft*, IMS, Universität Stuttgart, and SfS, Universität Tübingen.

Schmid, H.(2000), Lopar: Design and implementation, *Technical report*, IMS, Universität Stuttgart. Arbeitspapiere des Sonderforschungsbereichs 340.

Skut, W., Krenn, B., Brants, T. and Uszkoreit, H.(1997), An annotation scheme for free word order languages, *Proceedings of ANLP-97*, Washington, DC.

Telljohann, H., Hinrichs, E. W. and Kübler, S.(2003), *Stylebook for the German Treebank of Written German (TüBa-D/Z)*, Seminar für Sprachwissenschaft, Universität Tübingen, Tübingen.

Ule, T.(2003), Directed Treebank Refinement for PCFG parsing, *Proceedings of The Second Workshop on Treebanks and Linguistic Theories (TLT-2003)*, Vaxjö.

Ule, T. and Müller, F. H.(2004), KaRoPars: Ein System zur linguistischen Annotation großer Text-Korpora des Deutschen, *in* A. Mehler and H. Lobin (eds), *Automatische Textanalyse. Systeme und Methoden zur Annotation und Analyse natürlichsprachlicher Texte*, VS Verlag für Sozialwissenschaften, Opladen, pp. 185–202.

Veenstra, J., Müller, F. H. and Ule, T.(2002), Topological Fields Chunking for German, *Proceedings of CoNLL-2002*.