Parsing German with Latent Variable Grammars

Slav Petrov and Dan Klein
{petrov,klein}@cs.berkeley.edu
University of California at Berkeley
Berkeley, CA 94720

Abstract

We describe experiments on learning latent variable grammars for various German treebanks, using a language-agnostic statistical approach. In our method, a minimal initial grammar is hierarchically refined using an adaptive split-and-merge EM procedure, giving compact, accurate grammars. The learning procedure directly maximizes the likelihood of the training treebank, without the use of any language specific or linguistically constrained features. Nonetheless, the resulting grammars encode many linguistically interpretable patterns and give the best published parsing accuracies on three German treebanks.

1 Introduction

Probabilistic context-free grammars (PCFGs) underlie most high-performance parsers in one way or another (Collins, 1999; Charniak, 2000; Charniak and Johnson, 2005). However, as demonstrated in Charniak (1996) and Klein and Manning (2003), a PCFG which simply takes the empirical rules and probabilities off of a treebank does not perform well. This naive grammar is a poor one because its context-freedom assumptions are too strong in some ways (e.g. it assumes that subject and object NPs share the same distribution) and too weak in others (e.g. it assumes that long rewrites do not decompose into smaller steps). Therefore, a variety of techniques have been developed to both enrich and generalize the naive grammar, ranging from simple tree annotation and symbol splitting (Johnson, 1998; Klein and Manning, 2003) to full lexicalization and intricate smoothing (Collins, 1999; Charniak, 2000).

We view treebank parsing as the search for an optimally refined grammar consistent with a coarse training treebank. As a result, we begin with the provided evaluation symbols (such as NP, VP, etc.) but split them based on the statistical patterns in the training trees. A manual approach might take the symbol NP and subdivide it into one subsymbol NP^S for subjects and another subsymbol NP^VP for objects. However, rather than devising linguistically motivated features or splits, we take a fully automated approach, in which each symbol is split into unconstrained subsymbols. For example, NP would be split into NP-1 through NP-8. We use the Expectation-Maximization (EM) to then fit our split model to the observed trees; therein the various subsymbols will specialize in ways which may or may not correspond to our linguistic intuitions. This approach is relatively language independent, because the hidden subsymbols are induced automatically from the training trees based solely on data likelihood, though of course it is most applicable to strongly configurational languages.

In our experiments, we find that we can learn compact grammars that give the highest parsing accuracies in the 2008 Parsing German shared task. Our F1-scores of 69.8/84.0 (TIGER/TueBa-D/Z) are more than four points higher than those of the second best systems. Additionally, we investigate the patterns that are learned and show that the latent variable approach recovers linguistically interpretable phenomena. In our analysis, we pay particular attention to similarities and differences between

Proceedings of the ACL-08: HLT Workshop on Parsing German (PaGe-08), pages 33–39,
Columbus, Ohio, USA, June 2008.

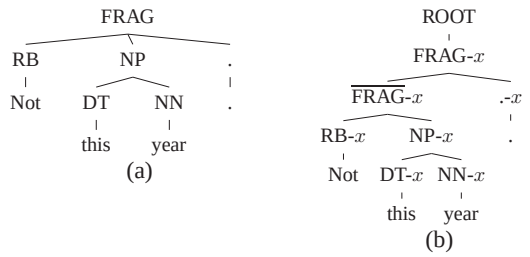

Figure 1: (a) The original tree. (b) The binarized tree with latent variables.

grammars learned from the two treebanks.

2 Latent Variable Parsing

In latent variable parsing (Matsuzaki et al., 2005; Prescher, 2005; Petrov et al., 2006), we learn rule probabilities on latent annotations that, when marginalized out, maximize the likelihood of the unannotated training trees. We use an automatic approach in which basic nonterminal symbols are alternately split and merged to maximize the likelihood of the training treebank.

In this section we briefly review the main ideas in latent variable parsing. This work has been previously published and we therefore provide only a short overview. For a more detailed exposition of the learning algorithm the reader is referred to Petrov et al. (2006). The corresponding inference procedure is described in detail in Petrov and Klein (2007). The parser, code, and trained models are available for download at `http://nlp.cs.berkeley.edu`.

2.1 Learning

Starting with a simple X-bar grammar, we use the Expectation-Maximization (EM) algorithm to learn a new grammar whose nonterminals are subsymbols of the original evaluation nonterminals. The X-bar grammar is created by binarizing the treebank trees; for each local tree rooted at an evaluation nonterminal X, we introduce a cascade of new nodes labeled $\overline{X}$ so that each node has at most two children, see Figure 1. This initialization is the absolute minimum starting grammar that distinguishes the evaluation nonterminals (and maintains separate grammars for each of them).

In Petrov et al. (2006) we show that a hierarchical split-and-merge strategy learns compact but accurate grammars, allocating subsymbols adaptively where they are most effective. Beginning with the baseline grammar, we repeatedly split and re-train the grammar. In each iteration, we initialize EM with the results of the previous round's grammar, splitting every previous symbol in two and adding a small amount of randomness (1%) to break the symmetry between the various subsymbols. Note that we split all nonterminal symbols, including the part-of-speech categories. While creating more latent annotations can increase accuracy, it can also lead to overfitting via oversplitting. Adding subsymbols divides grammar statistics into many bins, resulting in a tighter fit to the training data. At the same time, each bin has less support and therefore gives a less robust estimate of the grammar probabilities. At some point, the fit no longer generalizes, leading to overfitting.

To prevent oversplitting, we could measure the utility of splitting each latent annotation individually and then split the best ones first. However, not only is this impractical, requiring an entire training phase for each new split, but it assumes the contributions of multiple splits are independent. In fact, extra subsymbols may need to be added to several nonterminals before they can cooperate to pass information along the parse tree. This point is crucial to the success of our method: because all splits are fit simultaneously, local splits can chain together to propagate information non-locally. We therefore address oversplitting in the opposite direction; after training all splits, we measure for each one the loss in likelihood incurred by removing it. If this loss is small, the new annotation does not carry enough useful information and can be removed. Another advantage of evaluating post-hoc merges is that, unlike the likelihood gain from splitting, the likelihood loss from merging can be efficiently approximated.

To summarize, splitting provides an increasingly tight fit to the training data, while merging improves generalization and controls grammar size. In order to further overcome data fragmentation and overfitting, we also smooth our parameters along the split hierarchy. Smoothing allows us to add a larger number of annotations, each specializing in only a fraction of the data, without overfitting our training set.

2.2 Inference

At inference time, we want to use the learned grammar to efficiently and accurately compute a parse tree for a give sentence.

For efficiency, we employ a hierarchical coarse-to-fine inference scheme (Charniak et al., 1998; Charniak and Johnson, 2005; Petrov and Klein, 2007) which vastly improves inference time with no loss in test set accuracy. Our method considers the splitting history of the final grammar, projecting it onto its increasingly refined prior stages. For each such projection of the refined grammar, we estimate the projection's parameters from the source PCFG itself (rather than the original treebank), using techniques for infinite tree distributions and iterated fixpoint equations. We then rapidly pre-parse with each refinement stage in sequence, such that any item X:$[i,j]$ with sufficiently low posterior probability triggers the pruning of its further refined variants in all subsequent finer parses.

Our refined grammars G are over symbols of the form X-k where X is an evaluation symbol (such as *NP*) and k is some indicator of a subsymbol, which may encode something linguistic like a parent annotation context, but which is formally just an integer. G therefore induces a *derivation distribution* over trees labeled with split symbols. This distribution in turn induces a *parse distribution* over (projected) trees with unsplit evaluation symbols. We have several choices of how to select a tree given these posterior distributions over trees. Since computing the most likely parse tree is NP-complete (Sima'an, 1992), we settle for an approximation that allows us to (partially) sum out the latent annotation. In Petrov and Klein (2007) we relate this approximation to Goodman (1996)'s labeled brackets algorithm applied to rules and to Matsuzaki et al. (2005)'s sentence specific variational approximation. This procedure is substantially superior to simply erasing the latent annotations from the the Viterbi derivation.

2.3 Results

In Petrov and Klein (2007) we trained models for English, Chinese and German using the standard corpora and setups. We applied our latent variable model directly to each of the treebanks, without any language dependent modifications. Specifically, the same model hyperparameters (merging percentage and smoothing factor) were used in all experiments. Table 1 summarizes the results: automatically inducing latent structure is a technique that generalizes well across language boundaries and results in state of the art performance for Chinese and German. On English, the parser is outperformed by the reranked output of Charniak and Johnson (2005), but it outperforms their underlying lexicalized parser.

	≤ 40 words		all	
Parser	LP	LR	LP	LR
ENGLISH				
Charniak et al. (2005)	90.1	90.1	89.5	89.6
Petrov and Klein (2007)	**90.7**	**90.5**	**90.2**	**89.9**
ENGLISH (reranked)				
Charniak et al. (2005)	**92.4**	**91.6**	**91.8**	**91.0**
GERMAN (NEGRA)				
Dubey (2005)	F_1 76.3		-	
Petrov and Klein (2007)	**80.8**	**80.7**	**80.1**	**80.1**
CHINESE				
Chiang et al. (2002)	81.1	78.8	78.0	75.2
Petrov and Klein (2007)	**86.9**	**85.7**	**84.8**	**81.9**

Table 1: Our split-and-merge latent variable approach produces the best published parsing performance on many languages.

3 Experiments

We conducted experiments on the two treebanks provided for the 2008 Parsing German shared task. Both treebanks are annotated collections of German newspaper text, covering from similar topics. They are annotated with part-of-speech (POS) tags, morphological information, phrase structure, and grammatical functions. TueBa-D/Z additionally uses topological fields to describe fundamental word order restrictions in German clauses. However, the treebanks differ significantly in their annotation schemes: while TIGER relies on crossing branches to describe long distance relationships, TueBa-D/Z uses planar tree structures with designated labels that encode long distance relationships. Additionally, the annotation in TIGER is relatively flat on the phrasal level, while TueBa-D/Z annotates more internal phrase structure.

We used the standard splits into training and de-

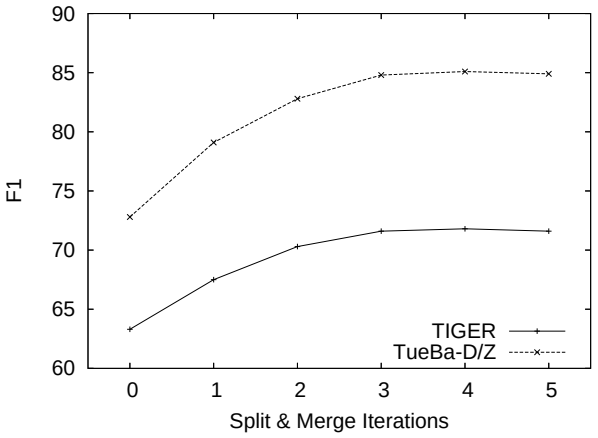

Figure 2: Parsing accuracy improves when the amount of latent annotation is increased.

velopment set, containing roughly 16,000 training trees and 1,600 development trees, respectively. All parsing figures in this section are on the development set, evaluating on constituents and grammatical functions using gold part-of-speech tags, unless noted otherwise. Note that even when we assume gold *evaluation* part-of-speech tags, we still assign probabilities to the different subsymbols of the provided evaluation tag. The parsing accuracies in the final results section are the official results of the 2008 Parsing German shared task.

3.1 Latent Annotation

As described in Section 2.1, we start with a minimal X-Bar grammar and learn increasingly refined grammars in a hierarchical split-and-merge fashion. We conjoined the constituency categories with their grammatical functions, creating initial categories like NP-PD and NP-OA which were further split automatically. Figure 2 shows how held-out accuracy improves when we add latent annotation. Our baseline grammars have low F1-scores (63.3/72.8, TIGER/TueBa-D/Z), but performance increases as the complexity of latent annotation increases. After four split-and-merge iterations, performance levels off. Interestingly, the gap in performance between the two treebanks increases from 9.5 to 13.4 F1-points. It appears that the latent variable approach is better suited for capturing the rich structure of the TueBa-D/Z treebank.

As languages vary in their phrase-internal headedness, we varied the binarization scheme, but, consistent with our experience in other languages, noticed little difference between right and left binarization. We also experimented with starting from a more constrained baseline by adding parent and sibling annotation. Adding initial structural annotation results in a higher baseline performance. However, since it fragments the grammar, adding latent annotation has a smaller effect, eventually resulting in poorer performance compared to starting from a simple X-Bar grammar. Essentially, the initial grammar is either mis- or oversplit to some degree.

	TIGER		TueBa-D/Z	
	F1	EX	F1	EX
Auto Tags	71.12	28.91	83.18	18.46
Gold Tags	71.74	34.04	85.10	20.98

Table 2: Parsing accuracies (F1-score and exact match) with gold POS tags and automatic POS tags. Many parse errors are due to incorrect tagging.

3.2 Part-of-speech tagging

When gold parts-of-speech are not assumed, many parse errors can be traced back to part-of-speech (POS) tagging errors. It is therefore interesting to investigate the influence of tagging errors on the overall parsing accuracy. For the shared task, we could assume gold POS tags: during inference we only allowed (and scored) the different subsymbols of the correct tags. However, this assumption cannot be made in a more realistic scenario, where we want to parse text from an unknown source. Table 2 compares the parsing performance with gold POS tags and with automatic tagging. While POS tagging errors have little influence on the TIGER treebank, tagging errors on TueBa-D/Z cause an substantial number of subsequent parse errors.

3.3 Two pass parsing

In the previous experiments, we conflated the phrasal categories and grammatical functions into single initial grammar symbol. An alternative is to first determine the categorical constituency structure and then to assign grammatical functions to the chosen constituents in a separate, second pass. To achieve this, we trained latent variable grammars for base constituency parsing by stripping off the

grammatical functions. After four rounds of split and merge training, these grammars achieve very good constituency accuracies of 85.1/94.1 F1-score (TIGER/TueBa-D/Z). For the second pass, we estimated (but did not split) X-Bar style grammars on the grammatical functions only. Fixing the constituency structure from the first pass, we used those to add grammatical functions. Unfortunately, this approach proved to be inferior to the unified, one pass approach, giving F1-scores of only 50.0/69.4 (TIGER/TueBa-D/Z). Presumably, the degradation can be attributed to the fact that grammatical functions model long-distance relations between the constituents, which can only be captured poorly by an unsplit, highly local X-bar style grammar.

3.4 Final Results

The final results of the shared task evaluation are shown in Table 3. These results were produced by a latent variable grammar that was trained for four split-and-merge iterations, starting from an X-Bar grammar over conjoined categorical/grammatical symbols, with a left-branching binarization. Our automatic latent variable approach serves better for German disambiguation than the competing approaches, despite its being very language agnostic.

4 Analysis

In this section, we examine the learned grammars, discussing what is learned. Because the grammatical functions significantly increase the number of base categories and make the grammars more difficult to examine, we show examples from grammars that were trained for categorical constituency parsing by initially stripping off all grammatical function annotations.

4.1 Lexical Splits

Since both treebanks use the same part-of-speech categories, it is easy to compare the learned POS subcategories. To better understand what is being learned, we selected two grammars after two split and merge iterations and examined the word distributions of the subcategories of various symbols. The three most likely words for a number of POS tags are shown in Table 4. Interestingly, the subcategories learned from the different treebanks exhibit very similar patterns. For example, in both cases, the nominal category (NE) has been split into subcategories for first and last names, abbreviations and places. The cardinal numbers (CARD) have been split into subcategories for years, spelled out numbers, and other numbers. There are often subcategories distinguishing sentence initial and sentence medial placement (KOND, PDAT, ART, APPR, etc.), as well as subcategories capturing case distinctions (PDAT, ART, etc.).

A quantitative way of analyzing the complexity of what is learned is to compare the number of subcategories that our split-and-merge procedure has allocated to each category. Table 5 shows the automatically determined number of subcategories for each POS tag. While many categories have been split into comparably many of subcategories, the POS tags in the TIGER treebank have in general been refined more heavily. This increased refinement can be explained by our merging criterion. We compute the loss in likelihood that would be incurred from removing a split, and we merge back the least useful splits. In this process, lexical and phrasal splits compete with each other. In TueBa-D/Z the phrasal categories have richer internal structure and therefore get split more heavily. As a consequence, the lexical categories are often relatively less refined at any given stage than in TIGER. Having different merging thresholds for the lexical and phrasal categories would eliminate this difference and we might expect the difference in lexical refinement to become less pronounced. Of course, because of the different underlying statistics in the two treebanks, we do not expect the number of subcategories to become exactly equal in any case.

4.2 Phrasal splits

Analyzing the phrasal splits is much more difficult, as the splits can model internal as well as external context (as well as combinations thereof) and, in general, several splits must be considered jointly before their patterning can be described. Furthermore, the two treebanks use different annotation standards and different constituent categories. Overall, the phrasal categories of the TueBa-D/Z treebank have been more heavily refined, in order to better capture the rich internal structures. In both treebanks, the most heavily split categories are the noun, verb and prepositional phrase categories (NP/NX,

	TIGER			TueBa-D/Z		
	LP	LR	F1	LP	LR	F1
Berkeley Parser	**69.23**	**70.41**	**69.81**	**83.91**	**84.04**	**83.97**
Växjö Parser	67.06	63.40	65.18	76.20	74.56	75.37
Stanford Parser	58.52	57.63	58.07	79.26	79.22	79.24

Table 3: Final test set results of the 2008 Parsing German shared task (labeled precision, labeled recall and F1-score) on both treebanks (including grammatical functions and using gold part-of-speech tags).

NE				NE			
Kohl	Klaus	SPD	Deutschland	Milosevic	Peter	K.	Berlin
Rabin	Helmut	USA	dpa	Müller	Wolfgang	W.	taz
Lafontaine	Peter	CDU	Bonn	Clinton	Klaus	de	Kosovo
CARD				CARD			
1996	zwei	000	zwei	1998	zwei	500	zwei
1994	drei	100	3	1999	drei	100	20
1991	vier	20	2	2000	fünf	20	18
KOND				KOND			
Und	und	sondern	und	Und	und	sondern	und
Doch	oder	aber	oder	Aber	oder	weder	Denn
Aber	aber	bis	sowie	Doch	aber	sowohl	oder
PDAT				PDAT			
Diese	dieser	diesem	-	Dieser	diese	diesem	dieser
Dieser	dieses	diese	-	Diese	dieser	dieser	diese
Dieses	diese	dieser	-	Dieses	dieses	diesen	dieses
ART				ART			
Die	der	der	die	Die	die	die	der
Der	des	den	der	die	Die	der	die
Das	Die	die	den	Der	das	den	den
APPR				APPR			
In	als	in	von	In	bis	in	von
Von	nach	von	in	Mit	Von	auf	in
Nach	vor	mit	für	Nach	Bis	mit	für
PDS				PDS			
Das	dessen	das	-	dem	dessen	das	Das
Dies	deren	dies	-	das	die	Das	das
Diese	die	diese	-	jene	denen	dies	diese

Table 4: The three most likely words for several part-of-speech (sub-)categories. The left column corresponds to the TIGER treebank the right column to the TueBa-D/Z treebank. Similar subcategories are learned for both treebanks.

POS	Ti	Tue
ADJA	32	17
NN	32	32
NE	31	32
ADV	30	15
ADJD	30	19
VVFIN	29	5
VVPP	29	4
APPR	25	24
VVINF	18	7
CARD	18	16
ART	10	7
PIS	9	14
PPER	9	2
PIDAT	-	9

POS	Ti	Tue
PIAT	8	7
VAFIN	8	3
KON	8	8
$[	7	11
PROAV	7	-
APPRART	6	5
$	6	2
PDS	5	5
PPOSAT	4	4
$.	4	5
PDAT	4	5
KOUS	4	3
VMFIN	4	1
PRELS	3	1

POS	Ti	Tue
VVIZU	3	2
VAINF	3	3
PTKNEG	3	1
FM	3	8
PWS	2	2
PWAV	2	5
XY	2	2
TRUNC	2	4
KOUI	2	1
PTKVZ	2	1
VAPP	2	2
KOKOM	2	5
PROP	-	2
VVIMP	1	1

POS	Ti	Tue
VAIMP	1	1
VMPP	1	2
PPOSS	1	1
PRELAT	1	1
NNE	1	-
APPO	1	1
PTKA	1	2
PTKANT	1	2
PWAT	1	2
PRF	1	1
PTKZU	1	1
APZR	1	1
VMINF	1	1
ITJ	1	2

Table 5: Automatically determined number of subcategories for the part-of-speech tags. The left column corresponds to the TIGER treebank the right column to the TueBa-D/Z treebank. Many categories are split in the same number of subcategories, but overall the TIGER categories have been more heavily refined.

PP/PX, VP/VX*) as well as the sentential categories (S/SIMPX). Categories that are rare or that have little internal structure, in contrast, have been split lightly or not at all.

5 Conclusions

We presented a series of experiments on parsing German with latent variable grammars. We showed that our latent variable approach is very well suited for parsing German, giving the best parsing figures on several different treebanks, despite being completely language independent. Additionally, we examined the learned grammars and showed examples illustrating the linguistically meaningful patterns that were learned. The parser, code, and models are available for download at `http://nlp.cs.berkeley.edu`.

References

E. Charniak and M. Johnson. 2005. Coarse-to-Fine N-Best Parsing and MaxEnt Discriminative Reranking. In *ACL'05*.

E. Charniak, S. Goldwater, and M. Johnson. 1998. Edge-based best-first chart parsing. 6^{th} *Workshop on Very Large Corpora*.

E. Charniak. 1996. Tree-bank grammars. In *AAAI '96*, pages 1031–1036.

E. Charniak. 2000. A maximum–entropy–inspired parser. In *NAACL '00*, pages 132–139.

D. Chiang and D. Bikel. 2002. Recovering latent information in treebanks. In *COLING '02*, pages 183–189.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, UPenn.

A. Dubey. 2005. What to do when lexicalization fails: parsing German with suffix analysis and smoothing. In *ACL '05*.

J. Goodman. 1996. Parsing algorithms and metrics. *ACL '96*.

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

D. Klein and C. Manning. 2003. Accurate unlexicalized parsing. In *ACL '03*, pages 423–430.

T. Matsuzaki, Y. Miyao, and J. Tsujii. 2005. Probabilistic CFG with latent annotations. In *ACL '05*.

S. Petrov and D. Klein. 2007. Improved inference for unlexicalized parsing. In *HLT-NAACL '07*.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *ACL '06*.

D. Prescher. 2005. Inducing head-driven PCFGs with latent heads: Refining a tree-bank grammar for parsing. In *ECML'05*.

K. Sima'an. 1992. Computatoinal complexity of probabilistic disambiguation. *Grammars*, 5:125–151.