Improved Inference for Unlexicalized Parsing

Slav Petrov and Dan Klein
Computer Science Division, EECS Department
University of California at Berkeley
Berkeley, CA 94720
{petrov,klein}@eecs.berkeley.edu

Abstract

We present several improvements to unlexicalized parsing with hierarchically state-split PCFGs. First, we present a novel coarse-to-fine method in which a grammar's own hierarchical projections are used for incremental pruning, including a method for efficiently computing projections of a grammar without a treebank. In our experiments, hierarchical pruning greatly accelerates parsing with no loss in empirical accuracy. Second, we compare various inference procedures for state-split PCFGs from the standpoint of risk minimization, paying particular attention to their practical tradeoffs. Finally, we present multilingual experiments which show that parsing with hierarchical state-splitting is fast and accurate in multiple languages and domains, even without any language-specific tuning.

1 Introduction

Treebank parsing comprises two problems: *learning*, in which we must select a model given a treebank, and *inference*, in which we must select a parse for a sentence given the learned model. Previous work has shown that high-quality unlexicalized PCFGs can be learned from a treebank, either by manual annotation (Klein and Manning, 2003) or automatic state splitting (Matsuzaki et al., 2005; Petrov et al., 2006). In particular, we demonstrated in Petrov et al. (2006) that a hierarchically split PCFG could exceed the accuracy of lexicalized PCFGs (Collins, 1999; Charniak and Johnson, 2005). However, many questions about inference with such split PCFGs remain open. In this work, we present

1. an effective method for pruning in split PCFGs
2. a comparison of objective functions for inference in split PCFGs,
3. experiments on automatic splitting for languages other than English.

In Sec. 3, we present a novel coarse-to-fine processing scheme for hierarchically split PCFGs. Our method considers the splitting history of the final grammar, projecting it onto its increasingly refined prior stages. For any projection of a grammar, we give a new method for efficiently estimating the projection's parameters from the source PCFG itself (rather than a treebank), using techniques for infinite tree distributions (Corazza and Satta, 2006) and iterated fixpoint equations. We then parse with each refinement, in sequence, much along the lines of Charniak et al. (2006), except with much more complex and automatically derived intermediate grammars. Thresholds are automatically tuned on held-out data, and the final system parses up to 100 times faster than the baseline PCFG parser, with no loss in test set accuracy.

In Sec. 4, we consider the well-known issue of inference objectives in split PCFGs. As in many model families (Steedman, 2000; Vijay-Shanker and Joshi, 1985), split PCFGs have a derivation / parse distinction. The split PCFG directly describes a generative model over derivations, but evaluation is sensitive only to the coarser treebank symbols. While the most probable parse problem is NP-complete (Sima'an, 1992), several approximate methods exist, including n-best reranking by parse likelihood, the labeled bracket algorithm of Goodman (1996), and a variational approximation introduced in Matsuzaki et al. (2005). We present experiments which explicitly minimize various evaluation risks over a candidate set using samples from the split PCFG, and relate those conditions to the existing non-sampling algorithms. We demonstrate that n-best reranking according to likelihood is superior for exact match, and that the non-reranking methods are superior for maximizing F_1. A specific contribution is to discuss the role of unary productions, which previous work has glossed over, but which is important in understanding why the various methods work as they do.

Finally, in Sec. 5, we learn state-split PCFGs for German and Chinese and examine out-of-domain performance for English. The learned grammars are compact and parsing is very quick in our multi-stage scheme. These grammars produce the highest test set parsing figures that we are aware of in each language, except for English for which non-local methods such as feature-based discriminative reranking are available (Charniak and Johnson, 2005).

2 Hierarchically Split PCFGs

We consider PCFG grammars which are derived from a raw treebank as in Petrov et al. (2006): A simple X-bar grammar is created by binarizing the treebank trees. We refer to this grammar as G_0. From this starting point, we iteratively refine the grammar in stages, as illustrated in Fig. 1. In each stage, all symbols are split in two, for example *DT* might become *DT-1* and *DT-2*. The refined grammar is estimated using a variant of the forward-backward algorithm (Matsuzaki et al., 2005). After a splitting stage, many splits are rolled back based on (an approximation to) their likelihood gain. This procedure gives an ontogeny of grammars G_i, where $G = G_n$ is the final grammar. Empirically, the gains on the English Penn treebank level off after 6 rounds. In Petrov et al. (2006), some simple smoothing is also shown to be effective. It is interesting to note that these grammars capture many of the "structural zeros" described by Mohri and Roark (2006) and pruning rules with probability below e^{-10} reduces the grammar size drastically without influencing parsing performance. Some of our methods and conclusions are relevant to all state-split grammars, such as Klein and Manning (2003) or Dreyer and Eisner (2006), while others apply most directly to the hierarchical case.

3 Search

When working with large grammars, it is standard to prune the search space in some way. In the case of lexicalized grammars, the unpruned chart often will not even fit in memory for long sentences. Several proven techniques exist. Collins (1999) combines a punctuation rule which eliminates many spans entirely, and then uses span-synchronous beams to prune in a bottom-up fashion. Charniak et al. (1998)

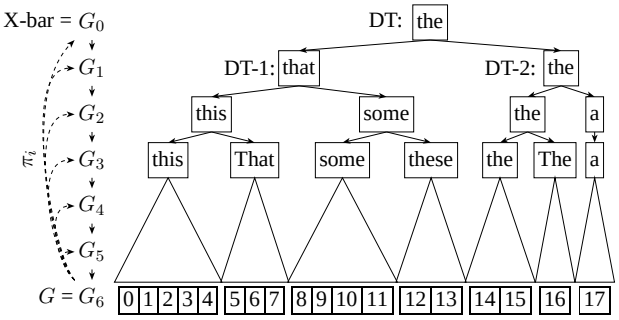

Figure 1: Hierarchical refinement proceeds top-down while projection recovers coarser grammars. The top word for the first refinements of the determiner tag (DT) is shown on the right.

introduces best-first parsing, in which a figure-of-merit prioritizes agenda processing. Most relevant to our work is Charniak and Johnson (2005) which uses a *pre-parse* phase to rapidly parse with a very coarse, unlexicalized treebank grammar. Any item $X{:}[i,j]$ with sufficiently low posterior probability in the pre-parse triggers the pruning of its lexical variants in a subsequent full parse.

3.1 Coarse-to-Fine Approaches

Charniak et al. (2006) introduces *multi-level coarse-to-fine* parsing, which extends the basic pre-parsing idea by adding more rounds of pruning. In their work, the extra pruning was with grammars even coarser than the raw treebank grammar, such as a grammar in which all nonterminals are collapsed. We propose a novel multi-stage coarse-to-fine method which is particularly natural for our hierarchically split grammar, but which is, in principle, applicable to any grammar. As in Charniak et al. (2006), we construct a sequence of increasingly refined grammars, reparsing with each refinement. The contributions of our method are that we derive sequences of refinements in a new way (Sec. 3.2), we consider refinements which are themselves complex, and, because our full grammar is not impossible to parse with, we automatically tune the pruning thresholds on held-out data.

3.2 Projection

In our method, which we call *hierarchical coarse-to-fine* parsing, we consider a sequence of PCFGs $G_0, G_1, \ldots G_n = G$, where each G_i is a refinement of the preceding grammar G_{i-1} and G is the full grammar of interest. Each grammar G_i is related to $G = G_n$ by a *projection* $\pi_{n \to i}$ or π_i for brevity. A

projection is a map from the non-terminal (including pre-terminal) symbols of G onto a reduced domain. A projection of grammar symbols induces a projection of rules and therefore entire non-weighted grammars (see Fig. 1).

In our case, we also require the projections to be sequentially compatible, so that $\pi_{i\to j} = \pi_{k\to j} \circ \pi_{i\to k}$. That is, each projection is itself a coarsening of the previous projections. In particular, we take the projection $\pi_{i\to j}$ to be the map that collapses split symbols in round i to their earlier identities in round j.

It is straightforward to take a projection π and map a CFG G to its induced projection $\pi(G)$. What is less obvious is how the probabilities associated with the rules of G should be mapped. In the case where $\pi(G)$ is more coarse than the treebank originally used to train G, and when that treebank is available, it is easy to project the treebank and directly estimate, say, the maximum-likelihood parameters for $\pi(G)$. This is the approach taken by Charniak et al. (2006), where they estimate what in our terms are projections of the raw treebank grammar from the treebank itself.

However, treebank estimation has several limitations. First, the treebank used to train G may not be available. Second, if the grammar G is heavily smoothed or otherwise regularized, its own distribution over trees may be far from that of the treebank. Third, the meanings of the split states can and do drift between splitting stages. Fourth, and most importantly, we may wish to project grammars for which treebank estimation is problematic, for example, grammars which are more refined than the observed treebank grammars. Our method effectively avoids all of these problems by rebuilding and refitting the pruning grammars on the fly from the final grammar.

3.2.1 Estimating Projected Grammars

Fortunately, there is a well worked-out notion of estimating a grammar from an infinite distribution over trees (Corazza and Satta, 2006). In particular, we can estimate parameters for a projected grammar $\pi(G)$ from the tree distribution induced by G (which can itself be estimated in any manner). The earliest work that we are aware of on estimating models from models in this way is that of Nederhof (2005), who considers the case of learning language models from other language models. Corazza and Satta (2006) extend these methods to the case of PCFGs and tree distributions.

The generalization of maximum likelihood estimation is to find the estimates for $\pi(G)$ with minimum KL divergence from the tree distribution induced by G. Since $\pi(G)$ is a grammar over coarser symbols, we fit $\pi(G)$ to the distribution G induces over π-projected trees: $P(\pi(T)|G)$. The proofs of the general case are given in Corazza and Satta (2006), but the resulting procedure is quite intuitive.

Given a (fully observed) treebank, the maximum-likelihood estimate for the probability of a rule $X \to YZ$ would simply be the ratio of the count of X to the count of the configuration $X \to YZ$. If we wish to find the estimate which has minimum divergence to an infinite distribution $P(T)$, we use the same formula, but the counts become expected counts:

$$P(X \to YZ) = \frac{E_{P(T)}[X \to YZ]}{E_{P(T)}[X]}$$

with unaries estimated similarly. In our specific case, X, Y, and Z are symbols in $\pi(G)$, and the expectations are taken over G's distribution of π-projected trees, $P(\pi(T)|G)$. We give two practical methods for obtaining these expectations below.

3.2.2 Calculating Projected Expectations

Concretely, we can now estimate the minimum divergence parameters of $\pi(G)$ for any projection π and PCFG G if we can calculate the expectations of the projected symbols and rules according to $P(\pi(T)|G)$. The simplest option is to sample trees T from G, project the samples, and take average counts off of these samples. In the limit, the counts will converge to the desired expectations, provided the grammar is proper. However, we can exploit the structure of our projections to obtain the desired expectations much more simply and efficiently.

First, consider the problem of calculating the expected counts of a symbol X in a tree distribution given by a grammar G, ignoring the issue of projection. These expected counts obey the following one-step equations (assuming a unique *root* symbol):

$$c(root) = 1$$

$$c(X) = \sum_{Y \to \alpha X \beta} P(\alpha X \beta | Y) c(Y)$$

Here, α, β, or both can be empty, and a rule $X \rightarrow \gamma$ appears in the sum once for each X it contains. In principle, this linear system can be solved in any way.[1] In our experiments, we solve this system iteratively, with the following recurrences:

$$c_0(X) \leftarrow \begin{cases} 1 & \text{if } X = root \\ 0 & \text{otherwise} \end{cases}$$

$$c_{i+1}(X) \leftarrow \sum_{Y \rightarrow \alpha X \beta} P(\alpha X \beta | Y) c_i(Y)$$

Note that, as in other iterative fixpoint methods, such as policy evaluation for Markov decision processes (Sutton and Barto, 1998), the quantities $c_k(X)$ have a useful interpretation as the expected counts ignoring nodes deeper than depth k (i.e. the roots are all the root symbol, so $c_0(root) = 1$). In our experiments this method converged within around 25 iterations; this is unsurprising, since the treebank contains few nodes deeper than 25 and our base grammar G seems to have captured this property.

Once we have the expected counts of symbols in G, the expected counts of their projections $X' = \pi(X)$ according to $P(\pi(T)|G)$ are given by $c(X') = \sum_{X:\pi(X)=X'} c(X)$. Rules can be estimated directly using similar recurrences, or given by one-step equations:

$$c(X \rightarrow \gamma) = c(X) P(\gamma | X)$$

This process very rapidly computes the estimates for a projection of a grammar (i.e. in a few seconds for our largest grammars), and is done once during initialization of the parser.

3.2.3 Hierarchical Projections

Recall that our final state-split grammars G come, by their construction process, with an ontogeny of grammars G_i where each grammar is a (partial) splitting of the preceding one. This gives us a natural chain of projections $\pi_{i \rightarrow j}$ which projects backwards along this ontogeny of grammars (see Fig. 1). Of course, training also gives us parameters for the grammars, but only the chain of projections is needed. Note that the projected estimates need not (and in general will not) recover the original parameters exactly, nor would we want them to. Instead they take into account any smoothing, substate drift, and so on which occurred by the final grammar.

Starting from the base grammar, we run the projection process for each stage in the sequence, calculating π_i (chained incremental projections would also be possible). For the remainder of the paper, except where noted otherwise, all coarser grammars' estimates are these reconstructions, rather than those originally learned.

3.3 Experiments

As demonstrated by Charniak et al. (2006) parsing times can be greatly reduced by pruning chart items that have low posterior probability under a simpler grammar. Charniak et al. (2006) pre-parse with a sequence of grammars which are coarser than (parent-annotated) treebank grammars. However, we also work with grammars which are already heavily split, up to half as split as the final grammar, because we found the computational cost for parsing with the simple X-bar grammar to be insignificant compared to the costs for parsing with more refined grammars.

For a final grammar $G = G_n$, we compute estimates for the n projections $G_{n-1}, \ldots, G_0$ =X-Bar, where $G_i = \pi_i(G)$ as described in the previous section. Additionally we project to a grammar G_{-1} in which all nonterminals, except for the preterminals, have been collapsed. During parsing, we start of by exhaustively computing the inside/outside scores with G_{-1}. At each stage, chart items with low posterior probability are removed from the chart, and we proceed to compute inside/outside scores with the next, more refined grammar, using the projections $\pi_{i \rightarrow i-1}$ to map between symbols in G_i and G_{i-1}. In each pass, we skip chart items whose projection into the previous stage had a probability below a stage-specific threshold, until we reach $G = G_n$ (after seven passes in our case). For G, we do not prune but instead return the minimum risk tree, as will be described in Sec. 4.

Fig. 2 shows the (unlabeled) bracket posteriors after each pass and demonstrates that most constructions can be ruled out by the simpler grammars, greatly reducing the amount of computation for the following passes. The pruning thresholds were empirically determined on a held out set by computing

[1] Whether or not the system has solutions depends on the parameters of the grammar. In particular, G may be improper, though the results of Chi (1999) imply that G will be proper if it is the maximum-likelihood estimate of a finite treebank.

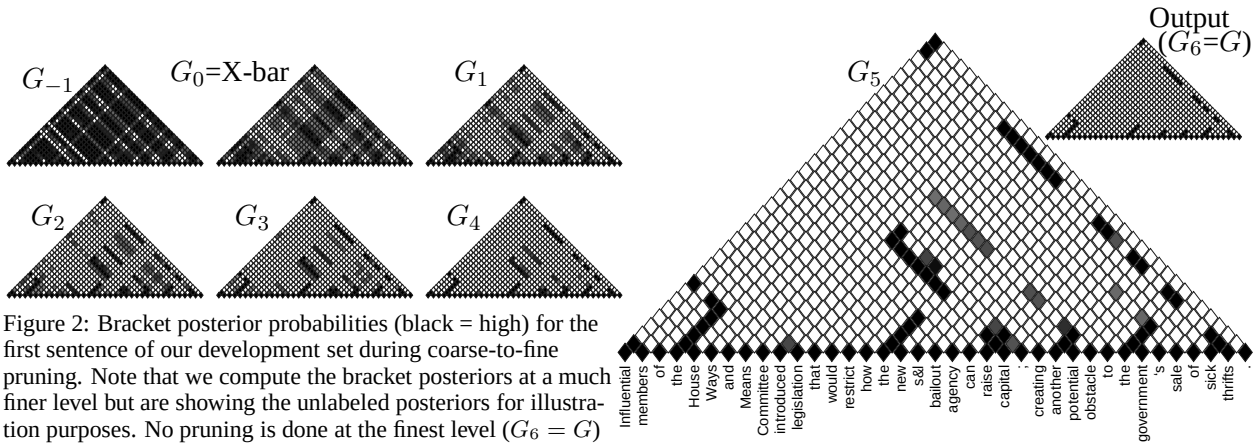

Figure 2: Bracket posterior probabilities (black = high) for the first sentence of our development set during coarse-to-fine pruning. Note that we compute the bracket posteriors at a much finer level but are showing the unlabeled posteriors for illustration purposes. No pruning is done at the finest level ($G_6 = G$) but the minimum risk tree is returned instead.

the most likely tree under G directly (without pruning) and then setting the highest pruning threshold for each stage that would not prune the optimal tree. This setting also caused no search errors on the test set. We found our projected grammar estimates to be at least equally well suited for pruning as the original grammar estimates which were learned during the hierarchical training. Tab. 1 shows the tremendous reduction in parsing time (all times are cumulative) and gives an overview over grammar sizes and parsing accuracies. In particular, in our Java implementation on a 3GHz processor, it is possible to parse the 1578 development set sentences (of length 40 or less) in less than 900 seconds with an F_1 of 91.2% (no search errors), or, by pruning more, in 600 seconds at 91.1%. For comparison, the Feb. 2006 release of the Charniak and Johnson (2005) parser runs in 1150 seconds on the same machine with an F_1 of 90.7%.

	G_0	G_2	G_4	G_6
Nonterminals	98	219	498	1140
Rules	3,700	19,600	126,100	531,200
No pruning	52 min	99 min	288 min	1612 min
X-bar pruning	8 min	14 min	30 min	111 min
C-to-F (no loss)	6 min	10 min	12 min	15 min
F_1 for above	64.8	85.2	89.7	91.2
C-to-F (lossy)	6 min	8 min	9 min	10 min
F_1 for above	64.3	84.7	89.4	91.1

Table 1: Grammar sizes, parsing times and accuracies for hierarchically split PCFGs with and without hierarchical coarse-to-fine parsing on our development set (1578 sentences with 40 or less words from section 22 of the Penn Treebank). For comparison the parser of Charniak and Johnson (2005) has an accuracy of F_1=90.7 and runs in 19 min on this set.

4 Objective Functions for Parsing

A split PCFG is a grammar G over symbols of the form X-k where X is an evaluation symbol (such as *NP*) and k is some indicator of a subcategory, such as a parent annotation. G induces a *derivation distribution* $P(T|G)$ over trees T labeled with split symbols. This distribution in turn induces a *parse distribution* $P(T'|G) = P(\pi(T)|G)$ over (projected) trees with unsplit evaluation symbols, where $P(T'|G) = \sum_{T:T'=\pi(T)} P(T|G)$. We now have several choices of how to select a tree given these posterior distributions over trees. In this section, we present experiments with the various options and explicitly relate them to parse risk minimization (Titov and Henderson, 2006).

The decision-theoretic approach to parsing would be to select the parse tree which minimizes our expected loss according to our beliefs:

$$T_P^* = \operatorname*{argmin}_{T_P} \sum_{T_T} P(T_T|w, G) L(T_P, T_T)$$

where T_T and T_P are "true" and predicted parse trees. Here, our loss is described by the function L whose first argument is the predicted parse tree and the second is the gold parse tree. Reasonable candidates for L include zero-one loss (exact match), precision, recall, F_1 (specifically EVALB here), and so on. Of course, the naive version of this process is intractable: we have to loop over all (pairs of) possible parses. Additionally, it requires parse likelihoods $P(T_P|w, G)$, which are tractable, but not trivial, to compute for split models. There are two options: limit the predictions to a small candidate set or choose methods for which dynamic programs exist.

For arbitrary loss functions, we can approximate the minimum-risk procedure by taking the min over only a set of *candidate parses* T_P. In some cases, each parse's expected risk can be evaluated in closed

Rule score: $r(\text{A} \rightarrow \text{B C}, i, k, j) = \sum_x \sum_y \sum_z \text{P}_{\text{OUT}}(A_x, i, j)\text{P}(\text{A}_x \rightarrow \text{B}_y\ \text{C}_z)\text{P}_{\text{IN}}(B_y, i, k)\text{P}_{\text{IN}}(C_y, k, j)$

VARIATIONAL:	$q(\text{A} \rightarrow \text{B C}, i, k, j) = \frac{r(\text{A} \rightarrow \text{B C}, i, k, j)}{\sum_x \text{P}_{\text{OUT}}(A_x, i, j)\text{P}_{\text{IN}}(A_x, i, j)}$	$T_G = \text{argmax}_T \prod_{e \in T} q(e)$
MAX-RULE-SUM:	$q(\text{A} \rightarrow \text{B C}, i, k, j) = \frac{r(\text{A} \rightarrow \text{B C}, i, k, j)}{\text{P}_{\text{IN}}(root, 0, n)}$	$T_G = \text{argmax}_T \sum_{e \in T} q(e)$
MAX-RULE-PRODUCT:	$q(\text{A} \rightarrow \text{B C}, i, k, j) = \frac{r(\text{A} \rightarrow \text{B C}, i, k, j)}{\text{P}_{\text{IN}}(root, 0, n)}$	$T_G = \text{argmax}_T \prod_{e \in T} q(e)$

Figure 3: Different objectives for parsing with posteriors, yielding comparable results. A, B, C are nonterminal symbols, x, y, z are latent annotations and i, j, k are between-word indices. Hence (A_x, i, j) denotes a constituent labeled with A_x spanning from i to j. Furthermore, we write $e = (\text{A} \rightarrow \text{B C}, i, j, k)$ for brevity.

form. Exact match (likelihood) has this property. In general, however, we can approximate the expectation with samples from $P(T|w, G)$. The method for sampling derivations of a PCFG is given in Finkel et al. (2006). It requires a single inside-outside computation per sentence and is then efficient per sample. Note that for split grammars, a posterior parse sample can be drawn by sampling a derivation and projecting away the substates.

Fig. 2 shows the results of the following experiment. We constructed 10-best lists from the full grammar G in Sec. 2 using the parser of Petrov et al. (2006). We then took the same grammar and extracted 500-sample lists using the method of Finkel et al. (2006). The minimum risk parse candidate was selected for various loss functions. As can be seen, in most cases, risk minimization reduces test-set loss of the relevant quantity. Exact match is problematic, however, because 500 samples is often too few to draw a match when a sentence has a very flat posterior, and so there are many all-way ties.[2] Since exact match permits a non-sampled calculation of the expected risk, we show this option as well, which is substantially superior. This experiment highlights that the correct procedure for exact match is to find the most probable parse.

An alternative approach to reranking candidate parses is to work with inference criteria which admit dynamic programming solutions. Fig. 3 shows three possible objective functions which use the easily obtained posterior marginals of the parse tree distribution. Interestingly, while they have fairly different decision theoretic motivations, their closed-form solutions are similar.

[2] 5,000 samples do not improve the numbers appreciably.

One option is to maximize likelihood in an approximate distribution. Matsuzaki et al. (2005) present a VARIATIONAL approach, which approximates the true posterior over parses by a cruder, but tractable sentence-specific one. In this approximate distribution there is no derivation / parse distinction and one can therefore optimize exact match by selecting the most likely derivation.

Instead of approximating the tree distribution we can use an objective function that decomposes along parse posteriors. The labeled brackets algorithm of Goodman (1996) has such an objective function. In its original formulation this algorithm maximizes the number of expected correct nodes, but instead we can use it to maximize the number of correct rules (the MAX-RULE-SUM algorithm). A worrying issue with this method is that it is ill-defined for grammars which allow infinite unary chains: there will be no finite minimum risk tree under recall loss (you can always reduce the risk by adding one more cycle). We implement MAX-RULE-SUM in a CNF-like grammar family where above each binary split is exactly one unary (possibly a self-loop). With this limitation, unary chains are not a problem. As might be expected, this criterion improves bracket measures at the expense of exact match.

We found it optimal to use a third approach, in which rule posteriors are multiplied instead of added. This corresponds to choosing the tree with greatest chance of having all rules correct, under the (incorrect) assumption that the rules correctness are independent. This MAX-RULE-PRODUCT algorithm does not need special treatment of infinite unary chains because it is optimizing a product rather than a sum. While these three methods yield

Objective	P	R	F1	EX
BEST DERIVATION				
Viterbi Derivation	89.6	89.4	89.5	37.4
RERANKING				
Random	87.6	87.7	87.7	16.4
Precision (sampled)	**91.1**	88.1	89.6	21.4
Recall (sampled)	88.2	**91.3**	89.7	21.5
F1 (sampled)	90.2	89.3	**89.8**	**27.2**
Exact (sampled)	89.5	89.5	89.5	25.8
Exact (non-sampled)	90.8	90.8	90.8	**41.7**
Exact/F1 (oracle)	95.3	94.4	95.0	63.9
DYNAMIC PROGRAMMING				
VARIATIONAL	90.7	90.9	90.8	**41.4**
MAX-RULE-SUM	90.5	**91.3**	90.9	40.4
MAX-RULE-PRODUCT	**91.2**	91.1	**91.2**	**41.4**

Table 2: A 10-best list from our best G can be reordered as to maximize a given objective either using samples or, under some restricting assumptions, in closed form.

very similar results (see Fig. 2), the MAX-RULE-PRODUCT algorithm consistently outperformed the other two.

Overall, the closed-form options were superior to the reranking ones, except on exact match, where the gains from correctly calculating the risk outweigh the losses from the truncation of the candidate set.

5 Multilingual Parsing

Most research on parsing has focused on English and parsing performance on other languages is generally significantly lower.[3] Recently, there have been some attempts to adapt parsers developed for English to other languages (Levy and Manning, 2003; Cowan and Collins, 2005). Adapting lexicalized parsers to other languages in not a trivial task as it requires at least the specification of head rules, and has had limited success. Adapting unlexicalized parsers appears to be equally difficult: Levy and Manning (2003) adapt the unlexicalized parser of Klein and Manning (2003) to Chinese, but even after significant efforts on choosing category splits, only modest performance gains are reported.

In contrast, automatically learned grammars like the one of Matsuzaki et al. (2005) and Petrov et al. (2006) require a treebank for training but no additional human input. One has therefore reason to believe that their performance will generalize better across languages than the performance of parsers that have been hand tailored to English.

[3]Of course, cross-linguistic comparison of results is complicated by differences in corpus annotation schemes and sizes, and differences in linguistic characteristics.

	ENGLISH (Marcus et al., 1993)	GERMAN (Skut et al., 1997)	CHINESE (Xue et al., 2002)
TrainSet	Sections 2-21	Sentences 1-18,602	Art. 1-270,400-1151
DevSet	Section 22	18,603-19,602	Articles 301-325
TestSet	Section 23	19,603-20,602	Articles 271-300

Table 3: Experimental setup.

5.1 Experiments

We trained models for English, Chinese and German using the standard corpora and splits as shown in Tab. 3. We applied our 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.

Tab. 4 shows that 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 only by the reranking parser of Charniak and Johnson (2005), which has access to a variety of features which cannot be captured by a generative model.

Space does not permit a thorough exposition of our analysis, but as in the case of English (Petrov et al., 2006), the learned subcategories exhibit interesting linguistic interpretations. In German, for example, the model learns subcategories for different cases and genders.

5.2 Corpus Variation

Related to cross language generalization is the generalization across domains for the same language. It is well known that a model trained on the Wall Street Journal loses significantly in performance when evaluated on the Brown Corpus (see Gildea (2001) for more details and the exact setup of their experiment, which we duplicated here). Recently McClosky et al. (2006) came to the conclusion that this performance drop is not due to overfitting the WSJ data. Fig. 4 shows the performance on the Brown corpus during hierarchical training. While the F_1 score on the WSJ is rising we observe a drop in performance after the 5th iteration, suggesting that some overfitting is occurring.

Parser	≤ 40 words LP	≤ 40 words LR	all LP	all LR
ENGLISH				
Charniak et al. (2005)	90.1	90.1	89.5	89.6
Petrov et al. (2006)	90.3	90.0	89.8	89.6
This Paper	**90.7**	**90.5**	**90.2**	**89.9**
ENGLISH (reranked)				
Charniak et al. (2005)[4]	**92.4**	**91.6**	**91.8**	**91.0**
GERMAN				
Dubey (2005)	F_1 76.3		-	
This Paper	**80.8**	**80.7**	**80.1**	**80.1**
CHINESE[5]				
Chiang et al. (2002)[6]	81.1	78.8	78.0	75.2
This Paper	**86.9**	**85.7**	**84.8**	**81.9**

Table 4: Our final test set parsing performance compared to the best previous work on English, German and Chinese.

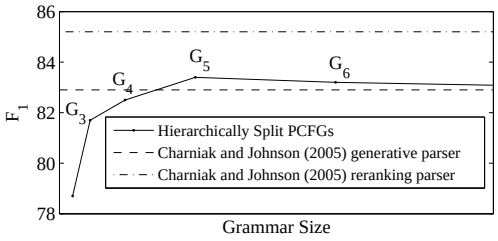

Figure 4: Parsing accuracy starts dropping after 5 training iterations on the Brown corpus, while it is improving on the WSJ, indicating overfitting.

6 Conclusions

The coarse-to-fine scheme presented here, in conjunction with the risk-appropriate parse selection methodology, allows fast, accurate parsing, in multiple languages and domains. For training, one needs only a raw context-free treebank and for decoding one needs only a final grammar, along with coarsening maps. The final parser is publicly available at `http://nlp.cs.berkeley.edu`.

Acknowledgments We would like to thank Eugene Charniak, Mark Johnson and Noah Smith for helpful discussions and comments.

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} *Wkshop on Very Large Corpora*.

E. Charniak, M. Johnson, et al. 2006. Multi-level coarse-to-fine PCFG Parsing. In *HLT-NAACL '06*.

Z. Chi. 1999. Statistical properties of probabilistic context-free grammars. In *Computational Linguistics*.

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

A. Corazza and G. Satta. 2006. Cross-entropy and estimation of probabilistic context-free grammars. In *HLT-NAACL '06*.

B. Cowan and M. Collins. 2005. Morphology and reranking for the statistical parsing of Spanish. In *HLT-EMNLP '05*.

M. Dreyer and J. Eisner. 2006. Better informed training of latent syntactic features. In *EMNLP '06*, pages 317–326.

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

J. Finkel, C. Manning, and A. Ng. 2006. Solving the problem of cascading errors: approximate Bayesian inference for lingusitic annotation pipelines. In *EMNLP '06*.

D. Gildea. 2001. Corpus variation and parser performance. *EMNLP '01*, pages 167–202.

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

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

R. Levy and C. Manning. 2003. Is it harder to parse Chinese, or the Chinese treebank? In *ACL '03*, pages 439–446.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. In *Computational Linguistics*.

T. Matsuzaki, Y. Miyao, and J. Tsujii. 2005. Probabilistic CFG with latent annotations. In *ACL '05*, pages 75–82.

D. McClosky, E. Charniak, and M. Johnson. 2006. Reranking and self-training for parser adaptation. In *COLING-ACL'06*.

M. Mohri and B. Roark. 2006. Probabilistic context-free grammar induction based on structural zeros. In *HLT-NAACL '06*.

M.-J. Nederhof. 2005. A general technique to train language models on language models. In *Computational Linguistics*.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *COLING-ACL '06*, pages 443–440.

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

W. Skut, B. Krenn, T. Brants, and H. Uszkoreit. 1997. An annotation scheme for free word order languages. In *Conference on Applied Natural Language Processing*.

M. Steedman. 2000. *The Syntactic Process.* The MIT Press, Cambridge, Massachusetts.

H. Sun and D. Jurafsky. 2004. Shallow semantic parsing of Chinese. In *HLT-NAACL '04*, pages 249–256.

R. Sutton and A. Barto. 1998. *Reinforcement Learning: An Introduction*. MIT Press.

I. Titov and J. Henderson. 2006. Loss minimization in parse reranking. In *EMNLP '06*, pages 560–567.

K. Vijay-Shanker and A. Joshi. 1985. Some computational properties of Tree Adjoining Grammars. In *ACL '85*.

N. Xue, F.-D. Chiou, and M. Palmer. 2002. Building a large scale annotated Chinese corpus. In *COLING '02*.

[5]This is the performance of the updated reranking parser available at http://www.cog.brown.edu/mj/software.htm

[6]Sun and Jurafsky (2004) report better performance, however they assume gold POS tags (*p.c.*).

[7]These results are obtained by training on a subset of the Chinese Treebank; our results on this subset are still superior.