Sparse Multi-Scale Grammars for Discriminative Latent Variable 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 a discriminative, latent variable approach to syntactic parsing in which rules exist at multiple scales of refinement. The model is formally a latent variable CRF grammar over trees, learned by iteratively splitting grammar productions (not categories). Different regions of the grammar are refined to different degrees, yielding grammars which are three orders of magnitude smaller than the single-scale baseline and 20 times smaller than the split-and-merge grammars of Petrov et al. (2006). In addition, our discriminative approach integrally admits features beyond local tree configurations. We present a multi-scale training method along with an efficient CKY-style dynamic program. On a variety of domains and languages, this method produces the best published parsing accuracies with the smallest reported grammars.

1 Introduction

In latent variable approaches to parsing (Matsuzaki et al., 2005; Petrov et al., 2006), one models an observed treebank of coarse *parse* trees using a grammar over more refined, but unobserved, *derivation* trees. The parse trees represent the desired output of the system, while the derivation trees represent the typically much more complex underlying syntactic processes. In recent years, latent variable methods have been shown to produce grammars which are as good as, or even better than, earlier parsing work (Collins, 1999; Charniak, 2000). In particular, in Petrov et al. (2006) we exhibited a very accurate category-splitting approach, in which a coarse initial grammar is refined by iteratively splitting each grammar category into two subcategories using the EM algorithm. Of course, each time the number of grammar categories is doubled, the number of binary productions is increased by a factor of eight. As a result, while our final grammars used few categories, the number of total active (non-zero) productions was still substantial (see Section 7). In addition, it is reasonable to assume that some generatively learned splits have little discriminative utility. In this paper, we present a discriminative approach which addresses both of these limitations.

We introduce *multi-scale* grammars, in which some productions reference fine categories, while others reference coarse categories (see Figure 2). We use the general framework of *hidden variable CRFs* (Lafferty et al., 2001; Koo and Collins, 2005), where gradient-based optimization maximizes the likelihood of the observed variables, here parse trees, summing over log-linearly scored derivations. With multi-scale grammars, it is natural to refine *productions* rather than categories. As a result, a category such as NP can be complex in some regions of the grammar while remaining simpler in other regions. Additionally, we exploit the flexibility of the discriminative framework both to improve the treatment of unknown words as well as to include *span features* (Taskar et al., 2004), giving the benefit of some input features integrally in our dynamic program. Our multi-scale grammars are 3 orders of magnitude smaller than the fully-split baseline grammar and 20 times smaller than the generative split-and-merge grammars of Petrov et al. (2006).

Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing, pages 867–876,
Honolulu, October 2008.

In addition, we exhibit the best parsing numbers on several metrics, for several domains and languages.

Discriminative parsing has been investigated before, such as in Johnson (2001), Clark and Curran (2004), Henderson (2004), Koo and Collins (2005), Turian et al. (2007), Finkel et al. (2008), and, most similarly, in Petrov and Klein (2008). However, in all of these cases, the final parsing performance fell short of the best generative models by several percentage points or only short sentences were used. Only in combination with a generative model was a discriminative component able to produce high parsing accuracies (Charniak and Johnson, 2005; Huang, 2008). Multi-scale grammars, in contrast, give higher accuracies using smaller grammars than previous work in this direction, outperforming top generative models in grammar size and in parsing accuracy.

2 Latent Variable Parsing

Treebanks are typically not annotated with fully detailed syntactic structure. Rather, they present only a coarse trace of the true underlying processes. As a result, learning a grammar for parsing requires the estimation of a more highly articulated model than the naive CFG embodied by such treebanks. A manual approach might take the category NP and subdivide it into one subcategory NP^S for subjects and another subcategory NP^VP for objects (Johnson, 1998; Klein and Manning, 2003). However, rather than devising linguistically motivated features or splits, latent variable parsing takes a fully automated approach, in which each symbol is split into unconstrained subcategories.

2.1 Latent Variable Grammars

Latent variable grammars augment the treebank trees with latent variables at each node. This creates a set of (exponentially many) *derivations* over split categories for each of the original *parse trees* over unsplit categories. For each observed category A we now have a set of latent subcategories A_x. For example, NP might be split into NP_1 through NP_8.

The parameters of the refined productions $A_x \rightarrow B_y\ C_z$, where A_x is a subcategory of A, B_y of B, and C_z of C, can then be estimated in various ways; past work has included both generative (Matsuzaki et al., 2005; Liang et al., 2007) and discriminative approaches (Petrov and Klein, 2008). We take the discriminative log-linear approach here. Note that the comparison is only between estimation methods, as Smith and Johnson (2007) show that the model classes are the same.

2.2 Log-Linear Latent Variable Grammars

In a log-linear latent variable grammar, each production $r = A_x \rightarrow B_y\ C_z$ is associated with a multiplicative weight ϕ_r (Johnson, 2001; Petrov and Klein, 2008) (sometimes we will use the log-weight θ_r when convenient). The probability of a derivation t of a sentence w is proportional to the product of the weights of its productions r:

$$P(t|w) \propto \prod_{r \in t} \phi_r$$

The score of a parse T is then the sum of the scores of its derivations:

$$P(T|w) = \sum_{t \in T} P(t|w)$$

3 Hierarchical Refinement

Grammar refinement becomes challenging when the number of subcategories is large. If each category is split into k subcategories, each (binary) production will be split into k^3. The resulting memory limitations alone can prevent the practical learning of highly split grammars (Matsuzaki et al., 2005). This issue was partially addressed in Petrov et al. (2006), where categories were repeatedly split and some splits were re-merged if the gains were too small. However, while the grammars are indeed compact at the (sub-)category level, they are still dense at the production level, which we address here.

As in Petrov et al. (2006), we arrange our subcategories into a hierarchy, as shown in Figure 1. In practice, the construction of the hierarchy is tightly coupled to a split-based learning process (see Section 5). We use the naming convention that an original category A becomes A_0 and A_1 in the first round; A_0 then becoming A_{00} and A_{01} in the second round, and so on. We will use $\hat{x} \succ x$ to indicate that the subscript or subcategory x is a refinement of $\hat{x}$.[1] We

[1] Conversely, $\hat{x}$ is a coarser version of x, or, in the language of Petrov and Klein (2007), $\hat{x}$ is a projection of x.

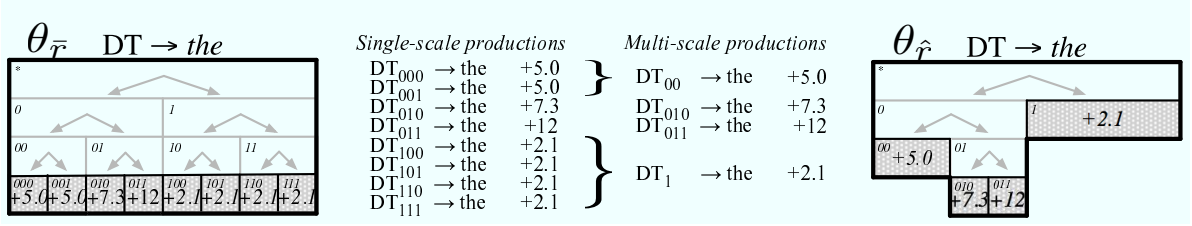

Figure 1: Multi-scale refinement of the $DT \rightarrow the$ production. The multi-scale grammar can be encoded much more compactly than the equally expressive single scale grammar by using only the shaded features along the fringe.

will also say that $\hat{x}$ dominates x, and $\overline{x}$ will refer to fully refined subcategories. The same terminology can be applied to (binary) productions, which split into eight refinements each time the subcategories are split in two.

The core observation leading to multi-scale grammars is that when we look at the refinements of a production, many are very similar in weight. It is therefore advantageous to record productions only at the level where they are distinct from their children in the hierarchy.

4 Multi-Scale Grammars

A multi-scale grammar is a grammar in which some productions reference fine categories, while others reference coarse categories. As an example, consider the multi-scale grammar in Figure 2, where the NP category has been split into two subcategories (NP_0, NP_1) to capture subject and object distinctions. Since *it* can occur in subject and object position, the production NP $\rightarrow$ *it* has remained unsplit. In contrast, in a single-scale grammar, two productions $NP_0 \rightarrow$ *it* and $NP_1 \rightarrow$ *it* would have been necessary. We use * as a wildcard, indicating that NP_* can combine with any other NP, while NP_1 can only combine with other NP_1. Whenever subcategories of different granularity are combined, the resulting constituent takes the more specific label.

In terms of its structure, a multi-scale grammar is a set of productions over varyingly refined symbols, where each production is associated with a weight. Consider the refinement of the production shown in Figure 1. The original unsplit production (at top) would naively be split into a tree of many subproductions (downward in the diagram) as the grammar categories are incrementally split. However, it may be that many of the fully refined productions share the same weights. This will be especially common in the present work, where we go out of our way to achieve it (see Section 5). For example, in Figure 1, the productions $DT_x \rightarrow$ *the* have the same weight for all categories DT_x which refine DT_1.[2] A multi-scale grammar can capture this behavior with just 4 productions, while the single-scale grammar has 8 productions. For binary productions the savings will of course be much higher.

In terms of its semantics, a multi-scale grammar is simply a compact encoding of a fully refined latent variable grammar, in which identically weighted refinements of productions have been collapsed to the coarsest possible scale. Therefore, rather than attempting to control the degree to which categories are split, multi-scale grammars simply encode productions at varying scales. It is hence natural to speak of refining productions, while considering the categories to exist at all degrees of refinement. Multi-scale grammars enable the use of coarse (even unsplit) categories in some regions of the grammar, while requiring very specific subcategories in others, as needed. As we will see in the following, this flexibility results in a tremendous reduction of grammar parameters, as well as improved parsing time, because the vast majority of productions end up only partially split.

Since a multi-scale grammar has productions which can refer to different levels of the category hierarchy, there must be constraints on their coherence. Specifically, for each fully refined production, exactly one of its dominating coarse productions must be in the grammar. More formally, the multi-scale grammar partitions the space of fully refined base rules such that each $\overline{r}$ maps to a unique

[2] We define dominating productions and refining productions analogously as for subcategories.

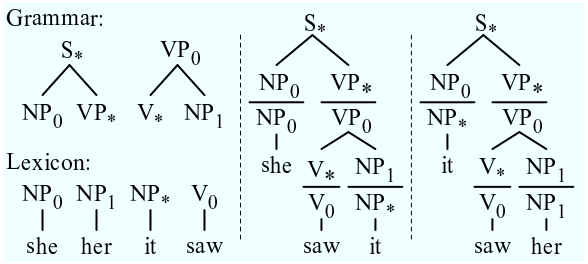

Figure 2: In multi-scale grammars, the categories exist at varying degrees of refinement. The grammar in this example enforces the correct usage of *she* and *her*, while allowing the use of *it* in both subject and object position.

dominating rule $\hat{r}$, and for all base rules $\overline{r}'$ such that $\hat{r} \succ \overline{r}'$, $\overline{r}'$ maps to $\hat{r}$ as well. This constraint is always satisfied if the multi-scale grammar consists of fringes of the production refinement hierarchies, indicated by the shading in Figure 1.

A multi-scale grammar straightforwardly assigns scores to derivations in the corresponding fully refined single scale grammar: simply map each refined derivation rule to its dominating abstraction in the multi-scale grammar and give it the corresponding weight. The fully refined grammar is therefore trivially (though not compactly) reconstructable from its multi-scale encoding.

It is possible to directly define a derivational semantics for multi-scale grammars which does not appeal to the underlying single scale grammar. However, in the present work, we use our multi-scale grammars only to compute expectations of the underlying grammars in an efficient, implicit way.

5 Learning Sparse Multi-Scale Grammars

We now consider how to discriminatively learn multi-scale grammars by iterative splitting productions. There are two main concerns. First, because multi-scale grammars are most effective when many productions share the same weight, sparsity is very desirable. In the present work, we exploit L_1-regularization, though other techniques such as structural zeros (Mohri and Roark, 2006) could also potentially be used. Second, training requires repeated parsing, so we use coarse-to-fine chart caching to greatly accelerate each iteration.

5.1 Hierarchical Training

We learn discriminative multi-scale grammars in an iterative fashion (see Figure 1). As in Petrov et al. (2006), we start with a simple X-bar grammar from an input treebank. The parameters θ of the grammar (production log-weights for now) are estimated in a log-linear framework by maximizing the penalized log conditional likelihood $\mathcal{L}_{cond} - R(\theta)$, where:

$$\mathcal{L}_{cond}(\theta) = \log \prod_i \mathrm{P}(T_i|w_i)$$

$$R(\theta) = \sum_r |\theta_r|$$

We directly optimize this non-convex objective function using a numerical gradient based method (LBFGS (Nocedal and Wright, 1999) in our implementation). To handle the non-diferentiability of the L_1-regularization term $R(\theta)$ we use the orthant-wise method of Andrew and Gao (2007). Fitting the log-linear model involves the following derivatives:

$$\frac{\partial \mathcal{L}_{cond}(\theta)}{\partial \theta_r} = \sum_i \Big(\mathbb{E}_\theta \left[f_r(t) | T_i \right] - \mathbb{E}_\theta [f_r(t) | w_i] \Big)$$

where the first term is the expected count f_r of a production r in derivations corresponding to the correct parse tree T_i and the second term is the expected count of the production in all derivations of the sentence w_i. Note that r may be of any scale. As we will show below, these expectations can be computed exactly using marginals from the chart of the inside/outside algorithm (Lari and Young, 1990).

Once the base grammar has been estimated, all categories are split in two, meaning that all binary productions are split in eight. When splitting an already refined grammar, we only split productions whose log-weight in the previous grammar deviates from zero.[3] This creates a refinement hierarchy over productions. Each newly split production r is given a unique feature, as well as inheriting the features of its parent productions $\hat{r} \succ r$:

$$\phi_r = \exp \Big(\sum_{\hat{r} \succ r} \theta_{\hat{r}} \Big)$$

The parent productions $\hat{r}$ are then removed from the grammar and the new features are fit as described

[3] L_1-regularization drives more than 95% of the feature weights to zero in each round.

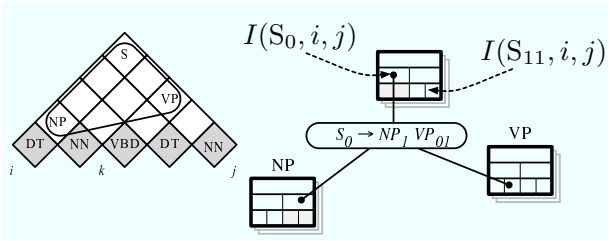

Figure 3: A multi-scale chart can be used to efficiently compute inside/outside scores using productions of varying specificity.

above. We detect that we have split a production too far when all child production features are driven to zero under L_1 regularization. In such cases, the children are collapsed to their parent production, which forms an entry in the multi-scale grammar.

5.2 Efficient Multi-Scale Inference

In order to compute the expected counts needed for training, we need to parse the training set, score all derivations and compute posteriors for all subcategories in the refinement hierarchy. The inside/outside algorithm (Lari and Young, 1990) is an efficient dynamic program for summing over derivations under a context-free grammar. It is fairly straightforward to adapt this algorithm to multi-scale grammars, allowing us to sum over an exponential number of derivations *without* explicitly reconstructing the underlying fully split grammar.

For single-scale latent variable grammars, the inside score $I(A_{\overline{x}}, i, j)$ of a fully refined category $A_{\overline{x}}$ spanning $\langle i, j\rangle$ is computed by summing over all possible productions $\overline{r} = A_{\overline{x}} \rightarrow B_{\overline{y}}\ C_{\overline{z}}$ with weight $\phi_{\overline{r}}$, spanning $\langle i, k\rangle$ and $\langle k, j\rangle$ respectively:[4]

$$I(A_{\overline{x}}, i, j) = \sum_{\overline{r}} \phi_{\overline{r}} \sum_{k} I(B_{\overline{y}}, i, k) I(C_{\overline{z}}, k, j)$$

Note that this involves summing over *all* relevant fully refined grammar productions.

The key quantities we will need are marginals of the form $I(A_x, i, j)$, the sum of the scores of all fully refined derivations rooted at any $A_{\overline{x}}$ dominated by A_x and spanning $\langle i, j\rangle$. We define these marginals in terms of the standard inside scores of the most refined subcategories $A_{\overline{x}}$:

$$I(A_x, i, j) = \sum_{\overline{x} \prec x} I(A_{\overline{x}}, i, j)$$

When working with multi-scale grammars, we expand the standard three-dimensional chart over spans and grammar categories to store the scores of all subcategories of the refinement hierarchy, as illustrated in Figure 3. This allows us to compute the scores more efficiently by summing only over rules $\hat{r} = A_{\hat{x}} \rightarrow B_{\hat{y}}\ C_{\hat{z}} \succ \overline{r}$:

$$\begin{aligned}
I(A_{\overline{x}}, i, j) &= \sum_{\hat{r}} \sum_{\overline{r} \prec \hat{r}} \phi_{\overline{r}} \sum_{k} I(B_{\overline{y}}, i, k) I(C_{\overline{z}}, k, j) \\
&= \sum_{\hat{r}} \phi_{\hat{r}} \sum_{\overline{r} \prec \hat{r}} \sum_{k} I(B_{\overline{y}}, i, k) I(C_{\overline{z}}, k, j) \\
&= \sum_{\hat{r}} \phi_{\hat{r}} \sum_{\overline{y} \prec \hat{y}} \sum_{\overline{z} \prec \hat{z}} \sum_{k} I(B_{\overline{y}}, i, k) I(C_{\overline{z}}, k, j) \\
&= \sum_{\hat{r}} \phi_{\hat{r}} \sum_{k} \sum_{\overline{y} \prec \hat{y}} I(B_{\overline{y}}, i, k) \sum_{\overline{z} \prec \hat{z}} I(C_{\overline{z}}, k, j) \\
&= \sum_{\hat{r}} \phi_{\hat{r}} \sum_{k} I(B_{\hat{y}}, i, k) I(C_{\hat{z}}, k, j)
\end{aligned}$$

Of course, some of the same quantities are computed repeatedly in the above equation and can be cached in order to obtain further efficiency gains. Due to space constraints we omit these details, and also the computation of the outside score, as well as the handling of unary productions.

5.3 Feature Count Approximations

Estimating discriminative grammars is challenging, as it requires repeatedly taking expectations over all parses of all sentences in the training set. To make this computation practical on large data sets, we use the same approach as Petrov and Klein (2008). Therein, the idea of coarse-to-fine parsing (Charniak et al., 1998) is extended to handle the repeated parsing of the same sentences. Rather than computing the entire coarse-to-fine history in every round of training, the pruning history is cached between training iterations, effectively avoiding the repeated calculation of similar quantities and allowing the efficient approximation of feature count expectations.

[4] These scores lack any probabilistic interpretation, but can be normalized to compute the necessary expectations for training (Petrov and Klein, 2008).

6 Additional Features

The discriminative framework gives us a convenient way of incorporating additional, overlapping features. We investigate two types of features: unknown word features (for predicting the part-of-speech tags of unknown or rare words) and span features (for determining constituent boundaries based on individual words and the overall sentence shape).

6.1 Unknown Word Features

Building a parser that can process arbitrary sentences requires the handling of previously unseen words. Typically, a classification of rare words into word classes is used (Collins, 1999). In such an approach, the word classes need to be manually defined *a priori*, for example based on discriminating word shape features (suffixes, prefixes, digits, etc.).

While this component of the parsing system is rarely talked about, its importance should not be underestimated: when using only one unknown word class, final parsing performance drops several percentage points. Some unknown word features are universal (e.g. digits, dashes), but most of them will be highly language dependent (prefixes, suffixes), making additional human expertise necessary for training a parser on a new language. It is therefore beneficial to automatically learn what the discriminating word shape features for a language are. The discriminative framework allows us to do that with ease. In our experiments we extract prefixes and suffixes of length ≤ 3 and add those features to words that occur 25 times or less in the training set. These unknown word features make the latent variable grammar learning process more language independent than in previous work.

6.2 Span Features

There are many features beyond local tree configurations which can enhance parsing discrimination; Charniak and Johnson (2005) presents a varied list. In reranking, one can incorporate any such features, of course, but even in our dynamic programming approach it is possible to include features that decompose along the dynamic program structure, as shown by Taskar et al. (2004). We use non-local *span features*, which condition on properties of input spans (Taskar et al., 2004). We illustrate our span features with the following example and the span $\langle 1, 4 \rangle$:

$$_0\ \text{“}\ _1[\ \textit{Yes}\ _2\ \text{”}\ _3,\ ]\ _4\ \textit{he}\ _5\ \textit{said}\ _6.\ _7$$

We first added the following lexical features:

- the first (*Yes*), last (*comma*), preceding (“) and following (*he*) words,
- the word pairs at the left edge ⟨“,*Yes*⟩, right edge ⟨*comma*,*he*⟩, inside border ⟨*Yes*,*comma*⟩ and outside border ⟨“,*he*⟩.

Lexical features were added for each span of length three or more. We used two groups of span features, one for natural constituents and one for synthetic ones.[5] We found this approach to work slightly better than anchoring the span features to particular constituent labels or having only one group.

We also added shape features, projecting the sentence to abstract shapes to capture global sentence structures. Punctuation shape replaces every non-punctuation word with `x` and then further collapses strings of `x` to `x+`. Our example becomes `#``x'',x+.#`, and the punctuation feature for our span is ```[x'',]x`. Capitalization shape projects the example sentence to `#.X..xx.#`, and `.[X..]x` for our span. Span features are a rich source of information and our experiments should be seen merely as an initial investigation of their effect in our system.

7 Experiments

We ran experiments on a variety of languages and corpora using the standard training and test splits, as described in Table 1. In each case, we start with a completely unannotated X-bar grammar, obtained from the raw treebank by a simple right-branching binarization scheme. We then train multi-scale grammars of increasing latent complexity as described in Section 5, directly incorporating the additional features from Section 6 into the training procedure. Hierarchical training starting from a raw treebank grammar and proceeding to our most refined grammars took three days in a parallel implementation using 8 CPUs. At testing time we marginalize out the hidden structure and extract the tree with the highest number of expected correct productions, as in Petrov and Klein (2007).

[5] Synthetic constituents are nodes that are introduced during binarization.

	Training Set	Dev. Set	Test Set
ENGLISH-WSJ (Marcus et al., 1993)	Sections 2-21	Section 22	Section 23
ENGLISH-BROWN (Francis et al. 2002)	see ENGLISH-WSJ	10% of the data[6]	10% of the the data[6]
FRENCH[7] (Abeille et al., 2000)	Sentences 1-18,609	Sentences 18,610-19,609	Sentences 19,609-20,610
GERMAN (Skut et al., 1997)	Sentences 1-18,602	Sentences 18,603-19,602	Sentences 19,603-20,602

Table 1: Corpora and standard experimental setups.

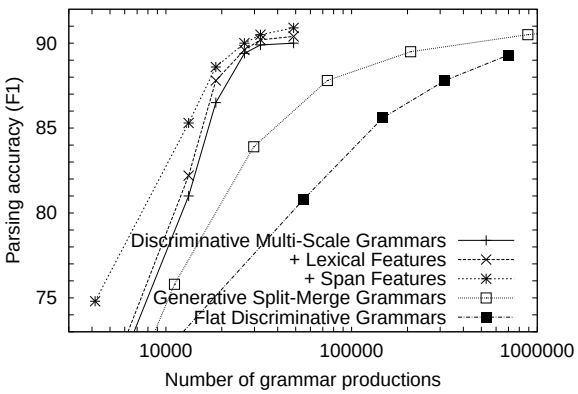

Figure 4: Discriminative multi-scale grammars give similar parsing accuracies as generative split-merge grammars, while using an order of magnitude fewer rules.

We compare to a baseline of discriminatively trained latent variable grammars (Petrov and Klein, 2008). We also compare our discriminative multi-scale grammars to their generative split-and-merge cousins, which have been shown to produce the state-of-the-art figures in terms of accuracy and efficiency on many corpora. For those comparisons we use the grammars from Petrov and Klein (2007).

7.1 Sparsity

One of the main motivations behind multi-scale grammars was to create compact grammars. Figure 4 shows parsing accuracies vs. grammar sizes. Focusing on the grammar size for now, we see that multi-scale grammars are extremely compact - even our most refined grammars have less than 50,000 active productions. This is 20 times smaller than the generative split-and-merge grammars, which use explicit category merging. The graph also shows that this compactness is due to controlling production sparsity, as the single-scale discriminative grammars are two orders of magnitude larger.

7.2 Accuracy

Figure 4 shows development set results for English. In terms of parsing accuracy, multi-scale grammars significantly outperform discriminatively trained single-scale latent variable grammars and perform on par with the generative split-and-merge grammars. The graph also shows that the unknown word and span features each add about 0.5% in final parsing accuracy. Note that the span features improve the performance of the unsplit baseline grammar by 8%, but not surprisingly their contribution gets smaller when the grammars get more refined. Section 8 contains an analysis of some of the learned features, as well as a comparison between discriminatively and generatively trained grammars.

7.3 Efficiency

Petrov and Klein (2007) demonstrates how the idea of coarse-to-fine parsing (Charniak et al., 1998; Charniak et al., 2006) can be used in the context of latent variable models. In coarse-to-fine parsing the sentence is rapidly pre-parsed with increasingly refined grammars, pruning away unlikely chart items in each pass. In their work the grammar is projected onto coarser versions, which are then used for pruning. Multi-scale grammars, in contrast, do not require projections. The refinement hierarchy is built in and can be used directly for coarse-to-fine pruning. Each production in the grammar is associated with a set of hierarchical features. To obtain a coarser version of a multi-scale grammar, one therefore simply limits which features in the refinement hierarchy can be accessed. In our experiments, we start by parsing with our coarsest grammar and allow an additional level of refinement at each stage of the pre-parsing. Compared to the generative parser of Petrov and Klein (2007), parsing with multi-scale grammars requires the evaluation of 29% fewer productions, decreasing the average parsing time per sentence by 36% to 0.36 sec/sentence.

[6] See Gildea (2001) for the exact setup.

[7] This setup contains only sentences without annotation errors, as in (Arun and Keller, 2005).

Parser	≤ 40 words F1	≤ 40 words EX	all F1	all EX
ENGLISH-WSJ				
Petrov and Klein (2008)	88.8	35.7	88.3	33.1
Charniak et al. (2005)	90.3	39.6	89.7	37.2
Petrov and Klein (2007)	**90.6**	39.1	**90.1**	37.1
This work w/o span features	89.7	39.6	89.2	37.2
This work w/ span features	90.0	**40.1**	89.4	**37.7**
ENGLISH-WSJ (reranked)				
Huang (2008)	**92.3**	**46.2**	**91.7**	**43.5**
ENGLISH-BROWN				
Charniak et al. (2005)	84.5	34.8	82.9	31.7
Petrov and Klein (2007)	84.9	34.5	83.7	31.2
This work w/o span features	85.3	35.6	84.3	32.1
This work w/ span features	**85.6**	**35.8**	**84.5**	**32.3**
ENGLISH-BROWN (reranked)				
Charniak et al. (2005)	**86.8**	**39.9**	**85.2**	**37.8**
FRENCH				
Arun and Keller (2005)	79.2	21.2	75.6	16.4
This Paper	**80.1**	**24.2**	**77.2**	**19.2**
GERMAN				
Petrov and Klein (2007)	80.8	40.8	80.1	39.1
This Paper	**81.5**	**45.2**	**80.7**	**43.9**

Table 2: Our final test set parsing accuracies compared to the best previous work on English, French and German.

7.4 Final Results

For each corpus we selected the grammar that gave the best performance on the development set to parse the final test set. Table 2 summarizes our final test set performance, showing that multi-scale grammars achieve state-of-the-art performance on most tasks. On WSJ-English, the discriminative grammars perform on par with the generative grammars of Petrov et al. (2006), falling slightly short in terms of F1, but having a higher exact match score. When trained on WSJ-English but tested on the Brown corpus, the discriminative grammars clearly outperform the generative grammars, suggesting that the highly regularized and extremely compact multi-scale grammars are less prone to overfitting. All those methods fall short of reranking parsers like Charniak and Johnson (2005) and Huang (2008), which, however, have access to many additional features, that cannot be used in our dynamic program.

When trained on the French and German treebanks, our multi-scale grammars achieve the best figures we are aware of, without any language specific modifications. This confirms that latent variable models are well suited for capturing the syntactic properties of a range of languages, and also shows that discriminative grammars are still effective when trained on smaller corpora.

8 Analysis

It can be illuminating to see the subcategories that are being learned by our discriminative multi-scale grammars and to compare them to generatively estimated latent variable grammars. Compared to the generative case, the lexical categories in the discriminative grammars are substantially less refined. For example, in the generative case, the nominal categories were fully refined, while in the discriminative case, fewer nominal clusters were heavily used. One reason for this can be seen by inspecting the first two-way split in the NNP tag. The generative model split into initial NNPs (*San, Wall*) and final NNPs (*Francisco, Street*). In contrast, the discriminative split was between organizational entities (*Stock, Exchange*) and other entity types (*September, New, York*). This constrast is unsurprising. Generative likelihood is advantaged by explaining lexical choice – *New* and *York* occur in very different slots. However, they convey the same information about the syntactic context above their base NP and are therefore treated the same, discriminatively, while the systematic attachment distinctions between temporals and named entities are more predictive.

Analyzing the syntactic and semantic patterns learned by the grammars shows similar trends. In Table 3 we compare the number of subcategories in the generative split-and-merge grammars to the average number of features per unsplit production with that phrasal category as head in our multi-scale grammars after 5 split (and merge) rounds. These quantities are inherently different: the number of features should be roughly cubic in the number of subcategories. However, we observe that the numbers are very close, indicating that, due to the sparsity of our productions, and the efficient multi-scale encoding, the number of grammar parameters grows linearly in the number of subcategories. Furthermore, while most categories have similar complexity in those two cases, the complexity of the two most refined phrasal categories are flipped. Generative grammars split NPs most highly, discrimina-

	NP	VP	PP	S	SBAR	ADJP	ADVP	QP	PRN
Generative subcategories	32	24	20	12	12	12	8	7	5
Discriminative production parameters	19	32	20	14	14	8	7	9	6

Table 3: Complexity of highly split phrasal categories in generative and discriminative grammars. Note that subcategories are compared to production parameters, indicating that the number of parameters grows cubicly in the number of subcategories for generative grammars, while growing linearly for multi-scale grammars.

	ENGLISH	GERMAN	FRENCH
Adjectives	-ous -ble -nth	-los -bar -ig	-ien -ble -ive
Nouns	-ion -en -cle	-tät -ung -rei	-té -eur -ges
Verbs	-ed -s	-st -eht	-ées -é
Adverbs	-ly	-mal	-ent
Numbers	-ty	-zig	—

Table 4: Automatically learned suffixes with the highest weights for different languages and part-of-speech tags.

tive grammars split the VP. This distinction seems to be because the complexity of VPs is more syntactic (e.g. complex subcategorization), while that of NPs is more lexical (noun choice is generally higher entropy than verb choice).

It is also interesting to examine the automatically learned word class features. Table 4 shows the suffixes with the highest weight for a few different categories across the three languages that we experimented with. The learning algorithm has selected discriminative suffixes that are typical derviational or inflectional morphemes in their respective languages. Note that the highest weighted suffixes will typically not correspond to the most common suffix in the word class, but to the most discriminative.

Finally, the span features also exhibit clear patterns. The highest scoring span features encourage the words between the last two punctuation marks to form a constituent (excluding the punctuation marks), for example `,[X+].` and `:[X+].` Words between quotation marks are also encouraged to form constituents: `''[X+]''` and `X[''X+'']X`. Span features can also discourage grouping words into constituents. The features with the highest negative weight involve single commas: `X[X,X+]`, and `X[X+,X+]X` and so on (indeed, such spans were structurally disallowed by the Collins (1999) parser).

9 Conclusions

Discriminatively trained multi-scale grammars give state-of-the-art parsing performance on a variety of languages and corpora. Grammar size is dramatically reduced compared to the baseline, as well as to methods like split-and-merge (Petrov et al., 2006). Because fewer parameters are estimated, multi-scale grammars may also be less prone to overfitting, as suggested by a cross-corpus evaluation experiment. Furthermore, the discriminative framework enables the seamless integration of additional, overlapping features, such as span features and unknown word features. Such features further improve parsing performance and make the latent variable grammars very language independent.

Our parser, along with trained grammars for a variety of languages, is available at http://nlp.cs.berkeley.edu.

References

A. Abeille, L. Clement, and A. Kinyon. 2000. Building a treebank for French. In *2nd International Conference on Language Resources and Evaluation*.

G. Andrew and J. Gao. 2007. Scalable training of L1-regularized log-linear models. In *ICML '07*.

A. Arun and F. Keller. 2005. Lexicalization in crosslinguistic probabilistic parsing: the case of french. In *ACL '05*.

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

E. Charniak. 2000. A maximum–entropy–inspired parser. In *NAACL '00*.

S. Clark and J. R. Curran. 2004. Parsing the WSJ using CCG and log-linear models. In *ACL '04*.

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

J. Finkel, A. Kleeman, and C. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *ACL '08*.

W. N. Francis and H. Kucera. 2002. Manual of information to accompany a standard corpus of present-day edited american english. In *TR, Brown University*.

D. Gildea. 2001. Corpus variation and parser performance. *EMNLP '01*.

J. Henderson. 2004. Discriminative training of a neural network statistical parser. In *ACL '04*.

L. Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *ACL '08*.

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

M. Johnson. 2001. Joint and conditional estimation of tagging and parsing models. In *ACL '01*.

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

T. Koo and M. Collins. 2005. Hidden-variable models for discriminative reranking. In *EMNLP '05*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *ICML '01*.

K. Lari and S. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*.

P. Liang, S. Petrov, M. I. Jordan, and D. Klein. 2007. The infinite PCFG using hierarchical Dirichlet processes. In *EMNLP '07*.

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*.

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

J. Nocedal and S. J. Wright. 1999. *Numerical Optimization*. Springer.

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

S. Petrov and D. Klein. 2008. Discriminative log-linear grammars with latent variables. In *NIPS '08*.

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

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

N. A. Smith and M. Johnson. 2007. Weighted and probabilistic context-free grammars are equally expressive. *Computational Lingusitics*.

B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. 2004. Max-margin parsing. In *EMNLP '04*.

J. Turian, B. Wellington, and I. D. Melamed. 2007. Scalable discriminative learning for natural language parsing and translation. In *NIPS '07*.