The Infinite PCFG using Hierarchical Dirichlet Processes

Percy Liang **Slav Petrov** **Michael I. Jordan** **Dan Klein**
Computer Science Division, EECS Department
University of California at Berkeley
Berkeley, CA 94720
{pliang, petrov, jordan, klein}@cs.berkeley.edu

Abstract

We present a nonparametric Bayesian model of tree structures based on the hierarchical Dirichlet process (HDP). Our HDP-PCFG model allows the complexity of the grammar to grow as more training data is available. In addition to presenting a fully Bayesian model for the PCFG, we also develop an efficient variational inference procedure. On synthetic data, we recover the correct grammar without having to specify its complexity in advance. We also show that our techniques can be applied to full-scale parsing applications by demonstrating its effectiveness in learning state-split grammars.

1 Introduction

Probabilistic context-free grammars (PCFGs) have been a core modeling technique for many aspects of linguistic structure, particularly syntactic phrase structure in treebank parsing (Charniak, 1996; Collins, 1999). An important question when learning PCFGs is how many grammar symbols to allocate to the learning algorithm based on the amount of available data.

The question of "how many clusters (symbols)?" has been tackled in the Bayesian nonparametrics literature via Dirichlet process (DP) mixture models (Antoniak, 1974). DP mixture models have since been extended to hierarchical Dirichlet processes (HDPs) and HDP-HMMs (Teh et al., 2006; Beal et al., 2002) and applied to many different types of clustering/induction problems in NLP (Johnson et al., 2006; Goldwater et al., 2006).

In this paper, we present the *hierarchical Dirichlet process PCFG* (HDP-PCFG). a nonparametric Bayesian model of syntactic tree structures based on Dirichlet processes. Specifically, an HDP-PCFG is defined to have an infinite number of symbols; the Dirichlet process (DP) prior penalizes the use of more symbols than are supported by the training data. Note that "nonparametric" does not mean "no parameters"; rather, it means that the effective number of parameters can grow adaptively as the amount of data increases, which is a desirable property of a learning algorithm.

As models increase in complexity, so does the uncertainty over parameter estimates. In this regime, point estimates are unreliable since they do not take into account the fact that there are different amounts of uncertainty in the various components of the parameters. The HDP-PCFG is a Bayesian model which naturally handles this uncertainty. We present an efficient variational inference algorithm for the HDP-PCFG based on a structured mean-field approximation of the true posterior over parameters. The algorithm is similar in form to EM and thus inherits its simplicity, modularity, and efficiency. Unlike EM, however, the algorithm is able to take the uncertainty of parameters into account and thus incorporate the DP prior.

Finally, we develop an extension of the HDP-PCFG for grammar refinement (HDP-PCFG-GR). Since treebanks generally consist of coarsely-labeled context-free tree structures, the maximum-likelihood treebank grammar is typically a poor model as it makes overly strong independence assumptions. As a result, many generative approaches to parsing construct refinements of the treebank grammar which are more suitable for the modeling task. Lexical methods split each pre-terminal symbol into many subsymbols, one for each word, and then focus on smoothing sparse lexical statis-

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 688–697, Prague, June 2007.

tics (Collins, 1999; Charniak, 2000). Unlexicalized methods refine the grammar in a more conservative fashion, splitting each non-terminal or pre-terminal symbol into a much smaller number of subsymbols (Klein and Manning, 2003; Matsuzaki et al., 2005; Petrov et al., 2006). We apply our HDP-PCFG-GR model to automatically learn the number of subsymbols for each symbol.

2 Models based on Dirichlet processes

At the heart of the HDP-PCFG is the Dirichlet process (DP) mixture model (Antoniak, 1974), which is the nonparametric Bayesian counterpart to the classical finite mixture model. In order to build up an understanding of the HDP-PCFG, we first review the Bayesian treatment of the finite mixture model (Section 2.1). We then consider the DP mixture model (Section 2.2) and use it as a building block for developing nonparametric structured versions of the HMM (Section 2.3) and PCFG (Section 2.4). Our presentation highlights the similarities between these models so that each step along this progression reflects only the key differences.

2.1 Bayesian finite mixture model

We begin by describing the Bayesian finite mixture model to establish basic notation that will carry over the more complex models we consider later.

Bayesian finite mixture model

$\boldsymbol{\beta} \sim$ Dirichlet$(\alpha, \dots, \alpha)$ [draw component probabilities]
For each component $z \in \{1, \dots, K\}$:
  $\phi_z \sim G_0$ [draw component parameters]

For each data point $i \in \{1, \dots, n\}$:
  $z_i \sim$ Multinomial$(\boldsymbol{\beta})$ [choose component]
  $x_i \sim F(\cdot; \phi_{z_i})$ [generate data point]

The model has K components whose prior distribution is specified by $\boldsymbol{\beta} = (\beta_1, \dots, \beta_K)$. The Dirichlet hyperparameter α controls how uniform this distribution is: as α increases, it becomes increasingly likely that the components have equal probability. For each mixture component $z \in \{1, \dots, K\}$, the parameters of the component ϕ_z are drawn from some prior G_0. Given the model parameters $(\boldsymbol{\beta}, \boldsymbol{\phi})$, the data points are generated i.i.d. by first choosing a component and then generating from a data model F parameterized by that component.

In document clustering, for example, each data point x_i is a document represented by its term-frequency vector. Each component (cluster) z has multinomial parameters ϕ_z which specifies a distribution $F(\cdot; \phi_z)$ over words. It is customary to use a conjugate Dirichlet prior $G_0 =$ Dirichlet$(\alpha', \dots, \alpha')$ over the multinomial parameters, which can be interpreted as adding $\alpha' - 1$ pseudocounts for each word.

2.2 DP mixture model

We now consider the extension of the Bayesian finite mixture model to a nonparametric Bayesian mixture model based on the Dirichlet process. We focus on the *stick-breaking representation* (Sethuraman, 1994) of the Dirichlet process instead of the stochastic process definition (Ferguson, 1973) or the Chinese restaurant process (Pitman, 2002). The stick-breaking representation captures the DP prior most explicitly and allows us to extend the finite mixture model with minimal changes. Later, it will enable us to readily define structured models in a form similar to their classical versions. Furthermore, an efficient variational inference algorithm can be developed in this representation (Section 2.6).

The key difference between the Bayesian finite mixture model and the DP mixture model is that the latter has a countably infinite number of mixture components while the former has a predefined K. Note that if we have an infinite number of mixture components, it no longer makes sense to consider a symmetric prior over the component probabilities; the prior over component probabilities must decay in some way. The stick-breaking distribution achieves this as follows. We write $\boldsymbol{\beta} \sim$ GEM(α) to mean that $\boldsymbol{\beta} = (\beta_1, \beta_2, \dots)$ is distributed according to the stick-breaking distribution. Here, the concentration parameter α controls the number of effective components. To draw $\boldsymbol{\beta} \sim$ GEM(α), we first generate a countably infinite collection of stick-breaking *proportions* $u_1, u_2, \dots$, where each $u_z \sim$ Beta$(1, \alpha)$. The stick-breaking weights $\boldsymbol{\beta}$ are then defined in terms of the stick proportions:

$$\beta_z = u_z \prod_{z' < z} (1 - u_{z'}). \tag{1}$$

The procedure for generating $\boldsymbol{\beta}$ can be viewed as iteratively breaking off remaining portions of a unit-

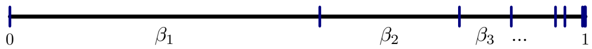


Figure 1: A sample $\boldsymbol{\beta} \sim \text{GEM}(1)$.

length stick (Figure 1). The component probabilities $\{\beta_z\}$ will decay exponentially in expectation, but there is always some probability of getting a smaller component before a larger one. The parameter α determines the decay of these probabilities: a larger α implies a slower decay and thus more components.

Given the component probabilities, the rest of the DP mixture model is identical to the finite mixture model:

DP mixture model

$\boldsymbol{\beta} \sim \text{GEM}(\alpha)$ [draw component probabilities]
For each component $z \in \{1, 2, \dots\}$:
  $\phi_z \sim G_0$ [draw component parameters]

For each data point $i \in \{1, \dots, n\}$:
  $z_i \sim \text{Multinomial}(\boldsymbol{\beta})$ [choose component]
  $x_i \sim F(\cdot; \phi_{z_i})$ [generate data point x_n]

2.3 HDP-HMM

The next stop on the way to the HDP-PCFG is the HDP hidden Markov model (HDP-HMM) (Beal et al., 2002; Teh et al., 2006). An HMM consists of a set of hidden states, where each state can be thought of as a mixture component. The parameters of the mixture component are the emission and transition parameters. The main aspect that distinguishes it from a flat finite mixture model is that the transition parameters themselves must specify a distribution over next states. Hence, we have not just one top-level mixture model over states, but also a collection of mixture models, one for each state.

In developing a nonparametric version of the HMM in which the number of states is infinite, we need to ensure that the transition mixture models of each state share a common inventory of possible next states. We can achieve this by tying these mixture models together using the hierarchical Dirichlet process (HDP) (Teh et al., 2006). The stick-breaking representation of an HDP is defined as follows: first, the top-level stick-breaking weights $\boldsymbol{\beta}$ are drawn according to the stick-breaking prior as before. Then, a new set of stick-breaking weights $\boldsymbol{\beta}'$ are generated according based on $\boldsymbol{\beta}$:

$$\boldsymbol{\beta}' \sim \text{DP}(\alpha', \boldsymbol{\beta}), \tag{2}$$

where the distribution of DP can be characterized in terms of the following finite partition property: for all partitions of the positive integers into sets $A_1, \dots, A_m$,

$$(\boldsymbol{\beta}'(A_1), \dots, \boldsymbol{\beta}'(A_m)) \sim \text{Dirichlet}\Big(\alpha'\boldsymbol{\beta}(A_1), \dots, \alpha'\boldsymbol{\beta}(A_m)\Big), \tag{3}$$

where $\boldsymbol{\beta}(A) = \sum_{k \in A} \beta_k$.[1] The resulting $\boldsymbol{\beta}'$ is another distribution over the positive integers whose similarity to $\boldsymbol{\beta}$ is controlled by a concentration parameter α'.

HDP-HMM

$\boldsymbol{\beta} \sim \text{GEM}(\alpha)$ [draw top-level state weights]
For each state $z \in \{1, 2, \dots\}$:
  $\phi_z^E \sim \text{Dirichlet}(\gamma)$ [draw emission parameters]
  $\phi_z^T \sim \text{DP}(\alpha', \beta)$ [draw transition parameters]

For each time step $i \in \{1, \dots, n\}$:
  $x_i \sim F(\cdot; \phi_{z_i}^E)$ [emit current observation]
  $z_{i+1} \sim \text{Multinomial}(\phi_{z_i}^T)$ [choose next state]

Each state z is associated with emission parameters ϕ_z^E. In addition, each z is also associated with transition parameters ϕ_z^T, which specify a distribution over next states. These transition parameters are drawn from a DP centered on the top-level stick-breaking weights $\boldsymbol{\beta}$ according to Equations (2) and (3). Assume that z_1 is always fixed to a special START state, so we do not need to generate it.

2.4 HDP-PCFG

We now present the HDP-PCFG, which is the focus of this paper. For simplicity, we consider Chomsky normal form (CNF) grammars, which has two types of rules: emissions and binary productions. We consider each grammar symbol as a mixture component whose parameters are the rule probabilities for that symbol. In general, we do not know the appropriate number of grammar symbols, so our strategy is to let the number of grammar symbols be infinite and place a DP prior over grammar symbols.

[1]Note that this property is a specific instance of the general stochastic process definition of Dirichlet processes.

HDP-PCFG

$\boldsymbol{\beta} \sim \text{GEM}(\alpha)$ [draw top-level symbol weights]

For each grammar symbol $z \in \{1, 2, \dots\}$:

- $\phi_z^T \sim \text{Dirichlet}(\alpha^T)$ [draw rule type parameters]
- $\phi_z^E \sim \text{Dirichlet}(\alpha^E)$ [draw emission parameters]
- $\phi_z^B \sim \text{DP}(\alpha^B, \boldsymbol{\beta}\boldsymbol{\beta}^T)$ [draw binary production parameters]

For each node i in the parse tree:

- $t_i \sim \text{Multinomial}(\phi_{z_i}^T)$ [choose rule type]
- If $t_i = \text{EMISSION}$:
  - $x_i \sim \text{Multinomial}(\phi_{z_i}^E)$ [emit terminal symbol]
- If $t_i = \text{BINARY-PRODUCTION}$:
  - $(z_{L(i)}, z_{R(i)}) \sim \text{Multinomial}(\phi_{z_i}^B)$ [generate children symbols]

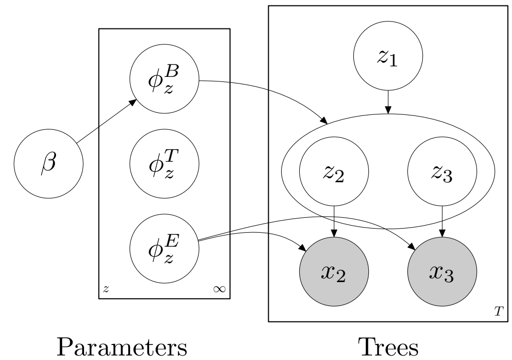


Figure 2: The definition and graphical model of the HDP-PCFG. Since parse trees have unknown structure, there is no convenient way of representing them in the visual language of traditional graphical models. Instead, we show a simple fixed example tree. Node 1 has two children, 2 and 3, each of which has one observed terminal child. We use $L(i)$ and $R(i)$ to denote the left and right children of node i.

In the HMM, the transition parameters of a state specify a distribution over single next states; similarly, the binary production parameters of a grammar symbol must specify a distribution over pairs of grammar symbols for its children. We adapt the HDP machinery to tie these binary production distributions together. The key difference is that now we must tie distributions over pairs of grammar symbols together via distributions over single grammar symbols.

Another difference is that in the HMM, at each time step, both a transition and a emission are made, whereas in the PCFG *either* a binary production or an emission is chosen. Therefore, each grammar symbol must also have a distribution over the type of rule to apply. In a CNF PCFG, there are only two types of rules, but this can be easily generalized to include unary productions, which we use for our parsing experiments.

To summarize, the parameters of each grammar symbol z consists of (1) a distribution over a finite number of rule types ϕ_z^T, (2) an emission distribution ϕ_z^E over terminal symbols, and (3) a binary production distribution ϕ_z^B over pairs of children grammar symbols. Figure 2 describes the model in detail.

Figure 3 shows the generation of the binary production distributions ϕ_z^B. We draw ϕ_z^B from a DP centered on $\boldsymbol{\beta}\boldsymbol{\beta}^T$, which is the product distribution over pairs of symbols. The result is a doubly-infinite matrix where most of the probability mass is concentrated in the upper left, just like the top-level distribution $\boldsymbol{\beta}\boldsymbol{\beta}^T$.

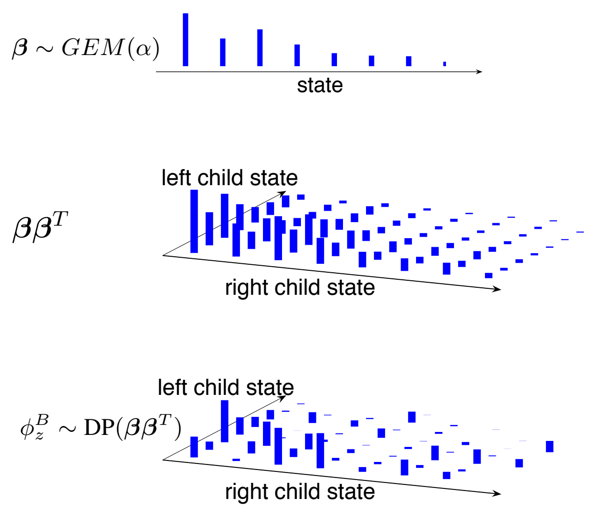


Figure 3: The generation of binary production probabilities given the top-level symbol probabilities $\boldsymbol{\beta}$. First, $\boldsymbol{\beta}$ is drawn from the stick-breaking prior, as in any DP-based model (a). Next, the outer-product $\boldsymbol{\beta}\boldsymbol{\beta}^T$ is formed, resulting in a doubly-infinite matrix matrix (b). We use this as the base distribution for generating the binary production distribution from a DP centered on $\boldsymbol{\beta}\boldsymbol{\beta}^T$ (c).

Note that we have replaced the general

G_0 and $F(\phi^E_{z_i})$ pair with Dirichlet(α^E) and Multinomial($\phi^E_{z_i}$) to specialize to natural language, but there is no difficulty in working with parse trees with arbitrary non-multinomial observations or more sophisticated word models.

In many natural language applications, there is a hard distinction between pre-terminal symbols (those that only emit a word) and non-terminal symbols (those that only rewrite as two non-terminal or pre-terminal symbols). This can be accomplished by letting $\alpha^T = (0, 0)$, which forces a draw ϕ^T_z to assign probability 1 to one rule type.

An alternative definition of an HDP-PCFG would be as follows: for each symbol z, draw a distribution over left child symbols $l_z \sim \mathrm{DP}(\boldsymbol{\beta})$ and an independent distribution over right child symbols $r_z \sim \mathrm{DP}(\boldsymbol{\beta})$. Then define the binary production distribution as their cross-product $\phi^B_z = l_z r_z^T$. This also yields a distribution over symbol pairs and hence defines a different type of nonparametric PCFG. This model is simpler and does not require any additional machinery beyond the HDP-HMM. However, the modeling assumptions imposed by this alternative are unappealing as they assume the left child and right child are independent given the parent, which is certainly not the case in natural language.

2.5 HDP-PCFG for grammar refinement

An important motivation for the HDP-PCFG is that of refining an existing treebank grammar to alleviate unrealistic independence assumptions and to improve parsing accuracy. In this scenario, the set of symbols is known, but we do not know how many subsymbols to allocate per symbol. We introduce the HDP-PCFG for grammar refinement (HDP-PCFG-GR), an extension of the HDP-PCFG, for this task.

The essential difference is that now we have a collection of HDP-PCFG models for each symbol $s \in S$, each one operating at the subsymbol level. While these HDP-PCFGs are independent in the prior, they are coupled through their interactions in the parse trees. For completeness, we have also included unary productions, which are essentially the PCFG counterpart of transitions in HMMs. Finally, since each node i in the parse tree involves a symbol-subsymbol pair (s_i, z_i), each subsymbol needs to specify a distribution over both child symbols and subsymbols. The former can be handled through a finite Dirichlet distribution since all symbols are known and observed, but the latter must be handled with the Dirichlet process machinery, since the number of subsymbols is unknown.

HDP-PCFG for grammar refinement (HDP-PCFG-GR)

For each symbol $s \in S$:
  $\boldsymbol{\beta}_s \sim \mathrm{GEM}(\alpha)$ [draw subsymbol weights]
  For each subsymbol $z \in \{1, 2, \dots\}$:
    $\phi^T_{sz} \sim \mathrm{Dirichlet}(\alpha^T)$ [draw rule type parameters]
    $\phi^E_{sz} \sim \mathrm{Dirichlet}(\alpha^E(s))$ [draw emission parameters]
    $\phi^u_{sz} \sim \mathrm{Dirichlet}(\alpha^u)$ [unary symbol productions]
    $\phi^b_{sz} \sim \mathrm{Dirichlet}(\alpha^b)$ [binary symbol productions]
    For each child symbol $s' \in S$:
      $\phi^U_{szs'} \sim \mathrm{DP}(\alpha^U, \boldsymbol{\beta}_{s'})$ [unary subsymbol prod.]
    For each pair of children symbols $(s', s'') \in S \times S$:
      $\phi^B_{szs's''} \sim \mathrm{DP}(\alpha^B, \boldsymbol{\beta}_{s'} \boldsymbol{\beta}^T_{s''})$ [binary subsymbol]

For each node i in the parse tree:
  $t_i \sim \mathrm{Multinomial}(\phi^T_{s_i z_i})$ [choose rule type]
  If t_i = EMISSION:
    $x_i \sim \mathrm{Multinomial}(\phi^E_{s_i z_i})$ [emit terminal symbol]
  If t_i = UNARY-PRODUCTION:
    $s_{L(i)} \sim \mathrm{Multinomial}(\phi^u_{s_i z_i})$ [generate child symbol]
    $z_{L(i)} \sim \mathrm{Multinomial}(\phi^U_{s_i z_i s_{L(i)}})$ [child subsymbol]
  If t_i = BINARY-PRODUCTION:
    $(s_{L(i)}, s_{R(i)}) \sim \mathrm{Mult}(\phi_{s_i z_i})$ [children symbols]
    $(z_{L(i)}, z_{R(i)}) \sim \mathrm{Mult}(\phi^B_{s_i z_i s_{L(i)} s_{R(i)}})$ [subsymbols]

2.6 Variational inference

We present an inference algorithm for the HDP-PCFG model described in Section 2.4, which can also be adapted to the HDP-PCFG-GR model with a bit more bookkeeping. Most previous inference algorithms for DP-based models involve sampling (Escobar and West, 1995; Teh et al., 2006). However, we chose to use variational inference (Blei and Jordan, 2005), which provides a fast deterministic alternative to sampling, hence avoiding issues of diagnosing convergence and aggregating samples. Furthermore, our variational inference algorithm establishes a strong link with past work on PCFG refinement and induction, which has traditionally employed the EM algorithm.

In EM, the E-step involves a dynamic program that exploits the Markov structure of the parse tree, and the M-step involves computing ratios based on expected counts extracted from the E-step. Our variational algorithm resembles the EM algorithm in form, but the ratios in the M-step are replaced with weights that reflect the uncertainty in parameter es-

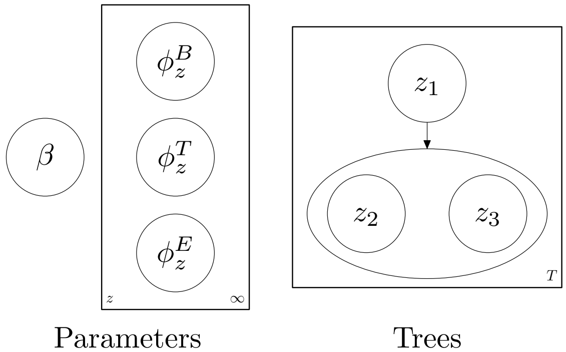


Figure 4: We approximate the true posterior p over parameters $\boldsymbol{\theta}$ and latent parse trees $\mathbf{z}$ using a structured mean-field distribution q, in which the distribution over parameters are completely factorized but the distribution over parse trees is unconstrained.

timates. Because of this procedural similarity, our method is able to exploit the desirable properties of EM such as simplicity, modularity, and efficiency.

2.7 Structured mean-field approximation

We denote parameters of the HDP-PCFG as $\boldsymbol{\theta} = (\boldsymbol{\beta}, \boldsymbol{\phi})$, where $\boldsymbol{\beta}$ denotes the top-level symbol probabilities and $\boldsymbol{\phi}$ denotes the rule probabilities. The hidden variables of the model are the training parse trees $\mathbf{z}$. We denote the observed sentences as $\mathbf{x}$.

The goal of Bayesian inference is to compute the posterior distribution $p(\boldsymbol{\theta}, \mathbf{z} \mid \mathbf{x})$. The central idea behind variational inference is to approximate this intractable posterior with a tractable approximation. In particular, we want to find the best distribution q^* as defined by

$$q^* \stackrel{\text{def}}{=} \operatorname*{argmin}_{q \in \mathcal{Q}} \mathrm{KL}(q(\boldsymbol{\theta}, \mathbf{z}) || p(\boldsymbol{\theta}, \mathbf{z} \mid \mathbf{x})), \quad (4)$$

where $\mathcal{Q}$ is a tractable subset of distributions. We use a *structured mean-field approximation*, meaning that we only consider distributions that factorize as follows (Figure 4):

$$\mathcal{Q} \stackrel{\text{def}}{=} \left\{ q(\mathbf{z}) q(\boldsymbol{\beta}) \prod_{z=1}^{K} q(\phi_z^T) q(\phi_z^E) q(\phi_z^B) \right\}. \quad (5)$$

We further restrict $q(\phi_z^T), q(\phi_z^E), q(\phi_z^B)$ to be Dirichlet distributions, but allow $q(\mathbf{z})$ to be any multinomial distribution. We constrain $q(\boldsymbol{\beta})$ to be a degenerate distribution truncated at K; i.e., $\beta_z = 0$ for $z > K$. While the posterior grammar does have an infinite number of symbols, the exponential decay of the DP prior ensures that most of the probability mass is contained in the first few symbols (Ishwaran and James, 2001).[2] While our variational approximation q is truncated, the actual PCFG model is not. As K increases, our approximation improves.

2.8 Coordinate-wise ascent

The optimization problem defined by Equation (4) is intractable and nonconvex, but we can use a simple coordinate-ascent algorithm that iteratively optimizes each factor of q in turn while holding the others fixed. The algorithm turns out to be similar in form to EM for an ordinary PCFG: optimizing $q(\mathbf{z})$ is the analogue of the E-step, and optimizing $q(\boldsymbol{\phi})$ is the analogue of the M-step; however, optimizing $q(\boldsymbol{\beta})$ has no analogue in EM. We summarize each of these updates below (see (Liang et al., 2007) for complete derivations).

Parse trees $q(\mathbf{z})$: The distribution over parse trees $q(\mathbf{z})$ can be summarized by the expected sufficient statistics (rule counts), which we denote as $C(z \to z_l\, z_r)$ for binary productions and $C(z \to x)$ for emissions. We can compute these expected counts using dynamic programming as in the E-step of EM.

While the classical E-step uses the current rule probabilities $\boldsymbol{\phi}$, our mean-field approximation involves an entire distribution $q(\boldsymbol{\phi})$. Fortunately, we can still handle this case by replacing each rule probability with a weight that summarizes the uncertainty over the rule probability as represented by q. We define this weight in the sequel.

It is a common perception that Bayesian inference is slow because one needs to compute integrals. Our mean-field inference algorithm is a counterexample: because we can represent uncertainty over rule probabilities with single numbers, much of the existing PCFG machinery based on EM can be modularly imported into the Bayesian framework.

Rule probabilities $q(\boldsymbol{\phi})$: For an ordinary PCFG, the M-step simply involves taking ratios of expected

[2]In particular, the variational distance between the stick-breaking distribution and the truncated version decreases exponentially as the truncation level K increases.

counts:

$$\phi_z^B(z_l, z_r) = \frac{C(z \to z_l\, z_r)}{C(z \to **)}. \qquad (6)$$

For the variational HDP-PCFG, the optimal $q(\boldsymbol{\phi})$ is given by the standard posterior update for Dirichlet distributions:[3]

$$q(\phi_z^B) = \text{Dirichlet}(\phi_z^B; \alpha^B \boldsymbol{\beta}\boldsymbol{\beta}^T + \vec{C}(z)), \qquad (7)$$

where $\vec{C}(z)$ is the matrix of counts of rules with left-hand side z. These distributions can then be summarized with *multinomial weights* which are the only necessary quantities for updating $q(\mathbf{z})$ in the next iteration:

$$\begin{aligned} W_z^B(z_l, z_r) &\stackrel{\text{def}}{=} \exp \mathbb{E}_q[\log \boldsymbol{\phi}_z^B(z_l, z_r)] &(8)\\ &= \frac{e^{\Psi(C(z\to z_l\, z_r)+\alpha^B \beta_{z_l}\beta_{z_r})}}{e^{\Psi(C(z\to **)+\alpha^B)}}, &(9) \end{aligned}$$

where $\Psi(\cdot)$ is the digamma function. The emission parameters can be defined similarly. Inspection of Equations (6) and (9) reveals that the only difference between the maximum likelihood and the mean-field update is that the latter applies the $\exp(\Psi(\cdot))$ function to the counts (Figure 5).

When the truncation K is large, $\alpha^B \beta_{z_l}\beta_{z_r}$ is near 0 for most right-hand sides (z_l, z_r), so $\exp(\Psi(\cdot))$ has the effect of downweighting counts. Since this subtraction affects large counts more than small counts, there is a rich-get-richer effect: rules that have already have large counts will be preferred.

Specifically, consider a set of rules with the same left-hand side. The weights for all these rules only differ in the numerator (Equation (9)), so applying $\exp(\Psi(\cdot))$ creates a *local preference* for right-hand sides with larger counts. Also note that the rule weights are not normalized; they always sum to at most one and are equal to one exactly when $q(\boldsymbol{\phi})$ is degenerate. This lack of normalization gives an extra degree of freedom not present in maximum likelihood estimation: it creates a *global preference* for left-hand sides that have larger total counts.

Top-level symbol probabilities $q(\boldsymbol{\beta})$: Recall that we restrict $q(\boldsymbol{\beta}) = \delta_{\boldsymbol{\beta}^*}(\boldsymbol{\beta})$, so optimizing $\boldsymbol{\beta}$ is equivalent to finding a single best $\boldsymbol{\beta}^*$. Unlike $q(\boldsymbol{\phi})$ and $q(\mathbf{z})$, there is no closed form expression for the optimal $\boldsymbol{\beta}^*$, and the objective function (Equation (4)) is not convex in $\boldsymbol{\beta}^*$. Nonetheless, we can apply a standard gradient projection method (Bertsekas, 1999) to improve $\boldsymbol{\beta}^*$ to a local maxima.

[3]Because we have truncated the top-level symbol weights, the DP prior on ϕ_z^B reduces to a finite Dirichlet distribution.

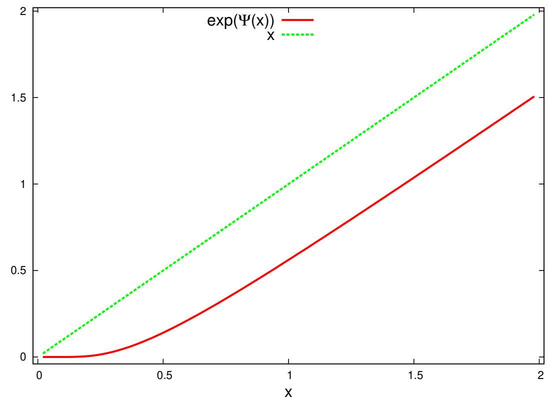


Figure 5: The $\exp(\Psi(\cdot))$ function, which is used in computing the multinomial weights for mean-field inference. It has the effect of reducing a larger fraction of small counts than large counts.

The part of the objective function in Equation (4) that depends on $\boldsymbol{\beta}^*$ is as follows:

$$\begin{aligned} L(\boldsymbol{\beta}^*) = \log \text{GEM}(\boldsymbol{\beta}^*; \alpha) + &\qquad (10)\\ \sum_{z=1}^{K} \mathbb{E}_q[\log \text{Dirichlet}(\phi_z^B; \alpha^B \boldsymbol{\beta}^* \boldsymbol{\beta}^{*T})]& \end{aligned}$$

See Liang et al. (2007) for the derivation of the gradient. In practice, this optimization has very little effect on performance. We suspect that this is because the objective function is dominated by $p(\mathbf{x} \mid \mathbf{z})$ and $p(\mathbf{z} \mid \boldsymbol{\phi})$, while the contribution of $p(\boldsymbol{\phi} \mid \boldsymbol{\beta})$ is minor.

3 Experiments

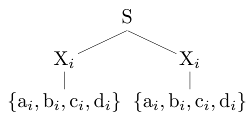
We now present an empirical evaluation of the HDP-PCFG(-GR) model and variational inference techniques. We first give an illustrative example of the ability of the HDP-PCFG to recover a known grammar and then present the results of experiments on large-scale treebank parsing.

3.1 Recovering a synthetic grammar

In this section, we show that the HDP-PCFG-GR can recover a simple grammar while a standard

S $\rightarrow \mathrm{X_1X_1} \mid \mathrm{X_2X_2} \mid \mathrm{X_3X_3} \mid \mathrm{X_4X_4}$
$\mathrm{X_1} \rightarrow \mathrm{a_1} \mid \mathrm{b_1} \mid \mathrm{c_1} \mid \mathrm{d_1}$
$\mathrm{X_2} \rightarrow \mathrm{a_2} \mid \mathrm{b_2} \mid \mathrm{c_2} \mid \mathrm{d_2}$
$\mathrm{X_3} \rightarrow \mathrm{a_3} \mid \mathrm{b_3} \mid \mathrm{c_3} \mid \mathrm{d_3}$
$\mathrm{X_4} \rightarrow \mathrm{a_4} \mid \mathrm{b_4} \mid \mathrm{c_4} \mid \mathrm{d_4}$

(a)



(b)

Figure 6: (a) A synthetic grammar with a uniform distribution over rules. (b) The grammar generates trees of the form shown on the right.

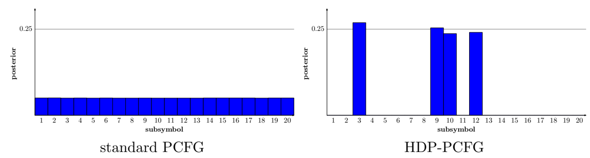


Figure 7: The posteriors over the subsymbols of the standard PCFG is roughly uniform, whereas the posteriors of the HDP-PCFG is concentrated on four subsymbols, which is the true number of symbols in the grammar.

PCFG fails to do so because it has no built-in control over grammar complexity. From the grammar in Figure 6, we generated 2000 trees. The two terminal symbols always have the same subscript, but we collapsed X_i to X in the training data. We trained the HDP-PCFG-GR, with truncation $K = 20$, for both S and X for 100 iterations. We set all hyperparameters to 1.

Figure 7 shows that the HDP-PCFG-GR recovers the original grammar, which contains only 4 subsymbols, leaving the other 16 subsymbols unused. The standard PCFG allocates all the subsymbols to fit the exact co-occurrence statistics of left and right terminals.

Recall that a rule weight, as defined in Equation (9), is analogous to a rule probability for standard PCFGs. We say a rule is *effective* if its weight is at least 10^{-6} and its left hand-side has posterior is also at least 10^{-6}. In general, rules with weight smaller than 10^{-6} can be safely pruned without affect parsing accuracy. The standard PCFG uses all 20 subsymbols of both S and X to explain the data, resulting in 8320 effective rules; in contrast, the HDP-PCFG uses only 4 subsymbols for X and 1 for S, resulting in only 68 effective rules. If the threshold is relaxed from 10^{-6} to 10^{-3}, then only 20 rules are effective, which corresponds exactly to the true grammar.

3.2 Parsing the Penn Treebank

In this section, we show that our variational HDP-PCFG can scale up to real-world data sets. We ran experiments on the Wall Street Journal (WSJ) portion of the Penn Treebank. We trained on sections 2–21, used section 24 for tuning hyperparameters, and tested on section 22.

We binarize the trees in the treebank as follows: for each non-terminal node with symbol X, we introduce a right-branching cascade of new nodes with symbol $\overline{X}$. The end result is that each node has at most two children. To cope with unknown words, we replace any word appearing fewer than 5 times in the training set with one of 50 unknown word tokens derived from 10 word-form features.

Our goal is to learn a refined grammar, where each symbol in the training set is split into K subsymbols. We compare an ordinary PCFG estimated with maximum likelihood (Matsuzaki et al., 2005) and the HDP-PCFG estimated using the variational inference algorithm described in Section 2.6.

To parse new sentences with a grammar, we compute the posterior distribution over rules at each span and extract the tree with the maximum expected correct number of rules (Petrov and Klein, 2007).

3.2.1 Hyperparameters

There are six hyperparameters in the HDP-PCFG-GR model, which we set in the following manner: $\alpha = 1$, $\alpha^T = 1$ (uniform distribution over unaries versus binaries), $\alpha^E = 1$ (uniform distribution over terminal words), $\alpha^u(s) = \alpha^b(s) = \frac{1}{N(s)}$, where $N(s)$ is the number of different unary (binary) right-hand sides of rules with left-hand side s in the treebank grammar. The two most important hyperparameters are α^U and α^B, which govern the sparsity of the right-hand side for unary and binary rules. We set $\alpha^U = \alpha^B$ although more performance could probably be gained by tuning these individually. It turns out that there is not a single α^B that works for all truncation levels, as shown in Table 1.

If the top-level distribution $\boldsymbol{\beta}$ is uniform, the value of α^B corresponding to a uniform prior over pairs of children subsymbols is K^2. Interestingly, the optimal α^B appears to be superlinear but subquadratic

truncation K	2	4	8	12	16	20
best α^B	16	12	20	28	48	80
uniform α^B	4	16	64	144	256	400

Table 1: For each truncation level, we report the α^B that yielded the highest F_1 score on the development set.

K	PCFG		PCFG (smoothed)		HDP-PCFG	
	F_1	Size	F_1	Size	F_1	Size
1	60.47	2558	60.36	2597	60.5	2557
2	69.53	3788	69.38	4614	71.08	4264
4	75.98	3141	77.11	12436	77.17	9710
8	74.32	4262	79.26	120598	79.15	50629
12	70.99	7297	78.8	160403	78.94	86386
16	66.99	19616	79.2	261444	78.24	131377
20	64.44	27593	79.27	369699	77.81	202767

Table 2: Shows development F_1 and grammar sizes (the number of effective rules) as we increase the truncation K.

in K. We used these values of α^B in the following experiments.

3.2.2 Results

The regime in which Bayesian inference is most important is when training data is scarce relative to the complexity of the model. We train on just section 2 of the Penn Treebank. Table 2 shows how the HDP-PCFG-GR can produce compact grammars that guard against overfitting. Without smoothing, ordinary PCFGs trained using EM improve as K increases but start to overfit around $K = 4$. Simple add-1.01 smoothing prevents overfitting but at the cost of a sharp increase in grammar sizes. The HDP-PCFG obtains comparable performance with a much smaller number of rules.

We also trained on sections 2–21 to demonstrate that our methods can scale up and achieve broadly comparable results to existing state-of-the-art parsers. When using a truncation level of $K = 16$, the standard PCFG with smoothing obtains an F_1 score of 88.36 using 706157 effective rules while the HDP-PCFG-GR obtains an F_1 score of 87.08 using 428375 effective rules. We expect to see greater benefits from the HDP-PCFG with a larger truncation level.

4 Related work

The question of how to select the appropriate grammar complexity has been studied in earlier work. It is well known that more complex models necessarily have higher likelihood and thus a penalty must be imposed for more complex grammars. Examples of such penalized likelihood procedures include Stolcke and Omohundro (1994), which used an asymptotic Bayesian model selection criterion and Petrov et al. (2006), which used a split-merge algorithm which procedurally determines when to switch between grammars of various complexities. These techniques are model selection techniques that use heuristics to choose among competing statistical models; in contrast, the HDP-PCFG relies on the Bayesian formalism to provide implicit control over model complexity within the framework of a single probabilistic model.

Johnson et al. (2006) also explored nonparametric grammars, but they do not give an inference algorithm for recursive grammars, e.g., grammars including rules of the form $A \rightarrow BC$ and $B \rightarrow DA$. Recursion is a crucial aspect of PCFGs and our inference algorithm does handle it. Finkel et al. (2007) independently developed another nonparametric model of grammars. Though their model is also based on hierarchical Dirichlet processes and is similar to ours, they present a different inference algorithm which is based on sampling. Kurihara and Sato (2004) and Kurihara and Sato (2006) applied variational inference to PCFGs. Their algorithm is similar to ours, but they did not consider nonparametric models.

5 Conclusion

We have presented the HDP-PCFG, a nonparametric Bayesian model for PCFGs, along with an efficient variational inference algorithm. While our primary contribution is the elucidation of the model and algorithm, we have also explored some important empirical properties of the HDP-PCFG and also demonstrated the potential of variational HDP-PCFGs on a full-scale parsing task.

References

C. E. Antoniak. 1974. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *Annals of Statistics*, 2:1152–1174.

M. Beal, Z. Ghahramani, and C. Rasmussen. 2002. The infinite hidden Markov model. In *Advances in Neural Information Processing Systems (NIPS)*, pages 577–584.

D. Bertsekas. 1999. *Nonlinear programming*.

D. Blei and M. I. Jordan. 2005. Variational inference for Dirichlet process mixtures. *Bayesian Analysis*, 1:121–144.

E. Charniak. 1996. Tree-bank grammars. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *North American Association for Computational Linguistics (NAACL)*, pages 132–139.

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

M. D. Escobar and M. West. 1995. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90:577–588.

T. S. Ferguson. 1973. A Bayesian analysis of some nonparametric problems. *Annals of Statistics*, 1:209–230.

J. R. Finkel, T. Grenager, and C. Manning. 2007. The infinite tree. In *Association for Computational Linguistics (ACL)*.

S. Goldwater, T. Griffiths, and M. Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *International Conference on Computational Linguistics and Association for Computational Linguistics (COLING/ACL)*.

H. Ishwaran and L. F. James. 2001. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 96:161–173.

M. Johnson, T. Griffiths, and S. Goldwater. 2006. Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models. In *Advances in Neural Information Processing Systems (NIPS)*.

D. Klein and C. Manning. 2003. Accurate unlexicalized parsing. In *Association for Computational Linguistics (ACL)*, pages 423–430.

K. Kurihara and T. Sato. 2004. An application of the variational Bayesian approach to probabilistic context-free grammars. In *International Joint Conference on Natural Language Processing Workshop Beyond Shallow Analyses*.

K. Kurihara and T. Sato. 2006. Variational Bayesian grammar induction for natural language. In *International Colloquium on Grammatical Inference*.

P. Liang, S. Petrov, M. I. Jordan, and D. Klein. 2007. Nonparametric PCFGs using Dirichlet processes. Technical report, Department of Statistics, University of California at Berkeley.

T. Matsuzaki, Y. Miyao, and J. Tsujii. 2005. Probabilistic CFG with latent annotations. In *Association for Computational Linguistics (ACL)*.

S. Petrov and D. Klein. 2007. Learning and inference for hierarchically split PCFGs. In *Human Language Technology and North American Association for Computational Linguistics (HLT/NAACL)*.

S. Petrov, L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *International Conference on Computational Linguistics and Association for Computational Linguistics (COLING/ACL)*.

J. Pitman. 2002. Combinatorial stochastic processes. Technical Report 621, Department of Statistics, University of California at Berkeley.

J. Sethuraman. 1994. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4:639–650.

A. Stolcke and S. Omohundro. 1994. Inducing probabilistic grammars by Bayesian model merging. In *Grammatical Inference and Applications*.

Y. W. Teh, M. I. Jordan, M. Beal, and D. Blei. 2006. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101:1566–1581.