Probabilistic Grammars and Hierarchical Dirichlet Processes

Percy Liang[*], Michael I. Jordan[*†], Dan Klein[*]

University of California at Berkeley

January 19, 2009

To appear in:

The Handbook of Applied Bayesian Analysis

Eds: Tony O'Hagan & Mike West
Oxford University Press

Abstract

Probabilistic context-free grammars (PCFGs) have played an important role in the modeling of syntax in natural language processing and other applications, but choosing the proper model complexity is often difficult. We present a nonparametric Bayesian generalization of the PCFG based on the hierarchical Dirichlet process (HDP). In our HDP-PCFG model, the effective complexity of the grammar can grow with increasing data. We describe an efficient variational inference algorithm for our model and present experiments on both a synthetic grammar induction task and a large-scale natural language parsing task.

Keywords: natural language processing, nonparametric Bayesian statistics, variational inference

[*]Computer Science Division, EECS Department, University of California at Berkeley, Berkeley CA 94720
[†]Department of Statistics, University of California at Berkeley, Berkeley CA 94720

1 Introduction

The field of natural language processing (NLP) aims to develop algorithms that allow computers to understand and generate natural language. The field emerged from computational linguistics, a field whose early history was shaped in part by a rejection of statistical approaches to language, where "statistical" at the time generally referred to simplistic Markovian models on observed sequences of words. Despite this unfavorable historical context, statistical approaches to NLP have been in ascendancy over the past decade (Manning and Schütze, 1999), driven in part by the availability of large corpora of text and other linguistic resources on the Internet, and driven in part by a growth in sophistication among NLP researchers regarding the scope of statistical modeling, particularly latent-variable modeling. The phenomenon of language itself is also responsible: language is replete with ambiguity, so it is inevitable that formal inferential methods should play a significant role in managing this ambiguity.

The majority of the work in statistical NLP has been non-Bayesian, but there is reason to believe that this is a historical accident. Indeed, despite the large corpora, sparse data problems abound to which hierarchical Bayesian methods seem well suited. Also, the conditional perspective of Bayesian statistics seems particularly appropriate for natural language—conditioning on the current sentence and the current context can provide precise inference despite the high degree of ambiguity.

In the current chapter, we discuss a Bayesian approach to the problem of *syntactic parsing* and the underlying problems of *grammar induction* and *grammar refinement*. The central object of study is the *parse tree*, an example of which is shown in Figure 1. A substantial amount of the syntactic structure and relational semantics of natural language sentences can be described using parse trees. These trees play a central role in a range of activities in modern NLP, including machine translation (Galley et al., 2004), semantic role extraction (Gildea and Jurafsky, 2002), and question answering (Hermjakob, 2001), just to name a few. From a statistical perspective, parse trees are an extremely rich class of objects, and our approach to capturing this class probabilistically will be to make use of tools from nonparametric Bayesian statistics.

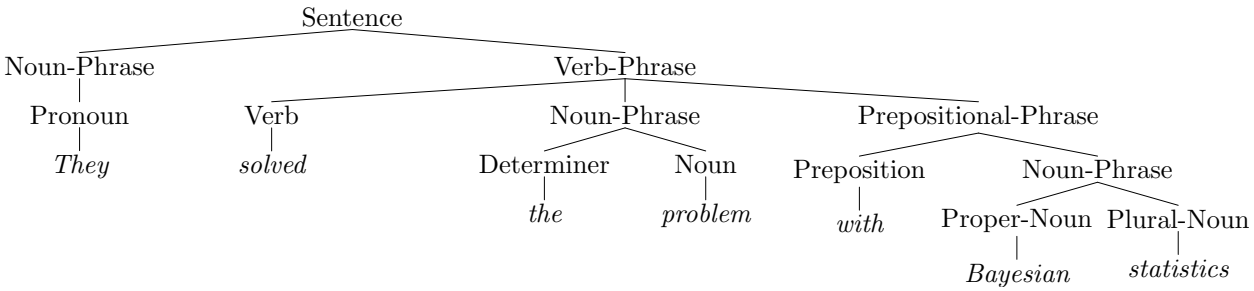


Figure 1: A parse tree for the sentence *They solved the problem with Bayesian statistics*.

It seems reasonable enough to model parse trees using context-free grammars (CFGs); indeed, this goal was the original motivation behind the development of the CFG formalism (Chomsky, 1956), and it remains a major focus of research on parsing to this day. Early work on NLP parsing concentrated on efficient algorithms for computing the set of all parses for a sentence under a given CFG. Unfortunately, as we have alluded to, natural language is highly ambiguous. In fact, the number of parses for a sentence grows exponentially with its length. As a result, systems which enumerated all possibilities were not useful in practice. Modern work on parsing has therefore turned to probabilistic models which place distributions over parse trees and probabilistic inference methods which focus on likely trees (Lari and Young, 1990).

The workhorse model family for probabilistic parsing is the family of *probabilistic context-free*

grammars (PCFGs),[1] which are probabilistic generalizations of CFGs and structural generalizations of hidden Markov models (HMMs). A PCFG (described formally in Section 1.1) is a branching process in which nodes iteratively rewrite from top to bottom, eventually terminating in dedicated *lexical* items, i.e., words. Each node is rewritten *independently* according to a multinomial distribution specific to that node's symbol. For example, a noun phrase frequently rewrites as a determiner followed by a noun (e.g., *the problem*).

Early work focused on *grammar induction* (also known as grammatical inference): estimating grammars directly from raw sentences without any other type of supervision (Carroll and Charniak, 1992). Grammar induction is an important scientific problem connecting cognitive science, linguistics, statistics, and even philosophy. A successful grammar induction system would have important implications for human language learning, and it would also be a valuable asset for being able to parse sentences with little human effort. However, a combination of model misspecification and local optima issues with the EM algorithm stymied these initial attempts. It turned out that it was necessary to impose more constraints on the tree structure (Pereira and Shabes, 1992).

Only with the advent of *treebanks* (Marcus et al., 1993)—hand-labeled collections of parse trees—were NLP researchers able to develop the first successful broad-coverage natural language parsers, but it still took a decade before the performance of the best parsers started to level off. In this supervised setting, maximum likelihood estimates for PCFGs have a simple closed-form solution: the rule probabilities of the PCFG are proportional to the counts of the associated grammar productions across all of the trees in the treebank (Charniak, 1996). However, such statistical grammars do not perform well for parsing. The problem is that treebanks contain only a handful of very coarse symbols, such as NP (noun phrase) and VP (verb phrase), so the conditional independences assumed by the PCFG over these coarse symbols are unrealistic. The true syntactic process is vastly more complex. For example, noun phrases can be subjects or objects, singular or plural, definite or indefinite, and so on. Similarly, verb phrases can be active or passive, transitive or intransitive, past or present, and so on. For a PCFG to adequately capture the true syntactic process, finer-grained grammar symbols are required. Much of the past decade of NLP parsing work can be seen as trying to optimally learn such fine-grained grammars from coarse treebanks.

Grammars can be refined in many ways. Some refinements are syntactically motivated. For example, if we augment each symbol with the symbol of its parent in the tree, we get symbols such as NP-VP, which represents direct object noun phrases, distinct from NP-S, which represents subject ones. This strategy is called *parent annotation* (Johnson, 1998) and can be extended (Klein and Manning, 2003). For the parse tree in Figure 1, if each node's symbol is augmented with the symbols of its parent and grandparent, a maximum likelihood grammar would only allow *they* to be produced under a noun phrase in subject position, disallowing ungrammatical sentences such as **The problem solved they with Bayesian statistics.*

Other refinements are semantically motivated. In many linguistic theories, each phrase is identified with a *head word*, which characterizes many of the important properties of the phrase. By augmenting the symbol of each node with the head word of the phrase under that node, we allow some degree of semantic plausibility to be captured by the parse tree. This process is called *lexicalization*, and was the basis for the first generation of practical treebank parsers (Collins, 1999; Charniak, 2000). Consider the sentence in Figure 1. It is actually ambiguous: did they use Bayesian statistics to solve the problem at hand (A) or did Bayesian statistics itself have a fundamental flaw which they resolved (B)? Though both are perfectly valid syntactically, (B) is implausible semantically, and we would like our model to prefer (A) over (B). If we lexicalize the verb phrase with *solved* (replace VP with VP-*solved*) and the preposition phrase with *statistics* (replace PP with PP-*statistics*),[2] we allow semantics to interact through the tree, yielding a better model of language that could pre-

[1] Also known as stochastic context-free grammars (SCFGs).

[2] According to standard NLP head conventions, the preposition *with* would be the head word of a prepositional phrase, but here we use a non-standard alternative to focus on the semantic properties of the phrase.

fer (A) over (B). Lexicalization produces a very rich model at the cost of multiplying the number of parameters by millions. In order to cope with the resulting problems of high-dimensionality, elaborate smoothing and parameter-tying methods were employed (Collins, 1999; Charniak, 2000), a recurrent theme in NLP.

Both parent annotation and lexicalization kept the grammar learning problem fully observed: Given the coarse trees in the treebank, the potential uncertainty resides in the choice of head words or parents, and these were typically propagated up the tree in deterministic ways. The only inferential problem remaining was to fit the grammar parameters, which reduces (in the point estimation setting generally adopted in statistical NLP) to counting and smoothing.

More recently, latent-variable models have been successfully employed for automatically refining treebank grammars (Matsuzaki et al., 2005; Petrov et al., 2006). In such approaches, each symbol is augmented with a latent cluster indicator variable and the marginal likelihood of the model is optimized. Rather than manually specifying the refinements, these latent-variable models let the data speak, and, empirically, the resulting refinements turn out to encode a mixture of syntactic and semantic information not captured by the coarse treebank symbols. The latent clusters allow for the modeling of long-range dependencies while keeping the number of parameters modest.

In this chapter, we address a fundamental question which underlies all of the previous work on parsing: what priors over PCFGs are appropriate? In particular, we know that we must trade off grammar complexity (the number of grammar symbols) against the amount of data present. As we get more data, more of the underlying grammatical processes can be adequately modeled. Past approaches to complexity control have considered minimum description length (Stolcke and Omohundro, 1994) and procedures for growing the grammar size heuristically (Petrov et al., 2006). While these procedures can be quite effective, we would like to pursue a nonparametric Bayesian approach to PCFGs so that we can state our assumptions about the problem of grammar growth in a coherent way. In particular, we define a nonparametric prior over PCFGs which allocates an unbounded number of symbols to the grammar through a Dirichlet process (Ferguson, 1973, 1974), then shares those symbols throughout the grammar using a Bayesian hierarchy of Dirichlet processes (Teh et al., 2006). We call the resulting model a *hierarchical Dirichlet process probabilistic context-free grammar* (HDP-PCFG). In this chapter, we present the formal probabilistic specification of the HDP-PCFG, algorithms for posterior inference under the HDP-PCFG, and experiments on grammar learning from large-scale corpora.

1.1 Probabilistic Context-Free Grammars (PCFGs)

Our HDP-PCFG model is based on probabilistic context-free grammars (PCFGs), which have been a core modeling technique for many aspects of syntactic structure (Charniak, 1996; Collins, 1999) as well as for problems in domains outside of natural language processing, including computer vision (Zhu and Mumford, 2006) and computational biology (Sakakibara, 2005; Dyrka and Nebel, 2007).

Formally, a PCFG is specified by the following:

- a set of *terminal* symbols Σ (the words in the sentence),
- a set of *nonterminal* symbols S,
- a designated *root* nonterminal symbol $\textsc{root} \in S$, and
- *rule probabilities* $\phi = (\phi_s(\gamma) : s \in S, \gamma \in \Sigma \cup (S \times S))$ with $\phi_s(\gamma) \geq 0$ and $\sum_\gamma \phi_s(\gamma) = 1$.

We restrict ourselves to rules $s \rightarrow \gamma$ that produce a right-hand side γ which is either a single terminal symbol ($\gamma \in \Sigma$) or a pair of nonterminal symbols ($\gamma \in S \times S$). Such a PCFG is said to be in *Chomsky normal form*. The restriction to Chomsky normal form is made without loss of generality;

it is straightforward to convert a rule with multiple children into a structure (e.g., a right-branching chain) in which each rule has at most two children.

A PCFG defines a distribution over sentences and parse trees via the following generative process: start at a root node with $s = \text{ROOT}$ and choose to apply rule $s \to \gamma$ with probability $\phi_s(\gamma)$; γ specifies the symbols of the children. For children with nonterminal symbols, recursively generate their subtrees. The process stops when all the leaves of the tree are terminals. Call the sequence of terminals the *yield*. More formally, a *parse tree* has a set of nonterminal nodes N along with the symbols corresponding to these nodes $s = (s_i \in S : i \in N)$. Let N_E denote the nodes having one terminal child and N_B denote the nodes having two nonterminal children. The tree structure is represented by $c = (c_j(i) : i \in N_B, j = 1, 2)$, where $c_j(i) \in N$ is the j-th child node of i from left to right. Let $z = (N, s, c)$ denote the parse tree and $x = (x_i : i \in N_E)$ denote the yield.

The joint probability of a parse tree z and its yield x is given by

$$p(x, z|\phi) = \prod_{i \in N_B} \phi_{s_i}\left(s_{c_1(i)}, s_{c_2(i)}\right) \prod_{i \in N_E} \phi_{s_i}(x_i). \tag{1}$$

PCFGs are similar to hidden Markov models (HMMs) and these similarities will guide our development of the HDP-PCFG. It is important to note at the outset, however, an important qualitative difference between HMMs and PCFGs. While the HMM can be represented as a graphical model (a Markovian graph in which the pattern of missing edges corresponds to assertions of conditional independence), the PCFG cannot. Conditioned on the structure of the parse tree (N, c), we have a graphical model over the symbols s, but the structure itself is a random object—we must run an algorithm to compute a probability distribution over these structures. As in the case of the forward-backward algorithm for HMMs, this algorithm is an efficient dynamic programming algorithm—it is referred to as the "inside-outside algorithm" and it runs in time cubic in length of the yield (Lari and Young, 1990). The inside-outside algorithm will play an important role in the inner loop of our posterior inference algorithm for the HDP-PCFG. Indeed, we will find it essential to design our model such that it can exploit the inside-outside algorithm.

Traditionally, PCFGs are defined with a fixed, finite number of nonterminals S, where the parameters ϕ are fit using (smoothed) maximum likelihood. The focus of this chapter is on developing a nonparametric version of the PCFG which allows S to be countably infinite and which performs approximate posterior inference over the set of nonterminal symbols and the set of parse trees. To define the HDP-PCFG and develop effective posterior inference algorithms for the HDP-PCFG, we need to bring several ingredients together—most notably the ability to generate new symbols and to tie together multiple usages of the same symbol on a parse tree (provided by the HDP), and the ability to efficiently compute probability distributions over parse trees (provided by the PCFG). Thus, the HDP-PCFG is a nonparametric Bayesian generalization of the PCFG, but it can also be viewed as a generalization along the Chomsky hierarchy, taking a nonparametric Markovian model (the HDP-HMM) to a nonparametric probabilistic grammar (the HDP-PCFG).

The rest of this chapter is organized as follows. Section 2 provides the probabilistic specification of the HDP-PCFG for grammar induction, and Section 3 extends this specification to an architecture appropriate for grammar refinement (the HDP-PCFG-GR). Section 4 describes an efficient variational method for approximate Bayesian inference in these models. Section 5 presents experiments: using the HDP-PCFG to induce a small grammar from raw text, and using the HDP-PCFG-GR to parse the Wall Street Journal, a standard large-scale dataset. For supplementary information, see the appendix, where we review DP-based models related to and leading up to the HDP-PCFG (Appendix A.1), discuss general issues related to approximate inference for these types of models (Appendix A.2), and provide some empirical intuition regarding the interaction between model and inference (Appendix A.3). The details of the variational inference algorithm are given in Appendix B.

HDP-PCFG

$\beta \sim \text{GEM}(\alpha)$ [draw top-level symbol probabilities]
For each grammar symbol $z \in \{1, 2, \dots\}$:
  $\phi_z^T \sim \text{Dir}(\alpha^T)$ [draw rule type parameters]
  $\phi_z^E \sim \text{Dir}(\alpha^E)$ [draw emission parameters]
  $\phi_z^B \sim \text{DP}(\alpha^B, \beta\beta^\top)$ [draw binary production parameters]

For each node i in the parse tree:
  $t_i \sim \text{Mult}(\phi_{z_i}^T)$ [choose rule type]
  If $t_i = \text{EMISSION}$:
    $x_i \sim \text{Mult}(\phi_{z_i}^E)$ [emit terminal symbol]
  If $t_i = \text{BINARY-PRODUCTION}$:
    $(z_{c_1(i)}, z_{c_2(i)}) \sim \text{Mult}(\phi_{z_i}^B)$ [generate child symbols]

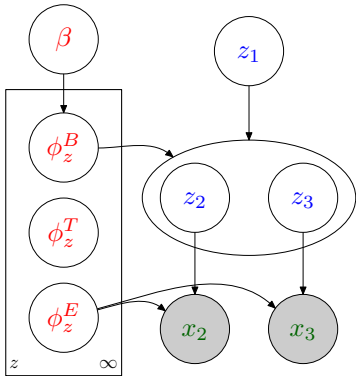


Figure 2: The probabilistic specification of the HDP-PCFG. We also present a graphical representation of the model. Drawing this graphical model assumes that the parse is known, which is *not* our assumption, so this representation should be viewed as simply suggestive of the HDP-PCFG. In particular, we show a simple fixed tree in which node z_1 has two children (z_2 and z_3), each of which has one observed terminal child.

2 The Hierarchical Dirichlet Process PCFG (HDP-PCFG)

At the core of the HDP-PCFG sits the Dirichlet process (DP) mixture model (Antoniak, 1974), a building block for a wide variety of nonparametric Bayesian models. The DP mixture model captures the basic notion of clustering which underlies symbol formation in the HDP-PCFG. From there, the HDP-PCFG involves several structural extensions of the DP mixture model. As a first stepping stone, consider hidden Markov models (HMMs), a dynamic generalization of mixture models, where clusters are linked structurally according to a Markov chain. To turn the HMM into a nonparametric Bayesian model, we use the hierarchical Dirichlet process (HDP), yielding the HDP-HMM, an HMM with a countably infinite state space. The HDP-PCFG differs from the HDP-HMM in that, roughly speaking, the HDP-HMM is a chain-structured model while HDP-PCFG is a tree-structured model. But it is important to remember that the tree structure in the HDP-PCFG is a random object over which inferences must be made. Another distinction is that rules can rewrite to two nonterminal symbols jointly. For this, we need to define DPs with base measures which are products of DPs, thereby adding another degree of complexity to the HDP machinery. For a gradual introduction, see Appendix A.1, where we walk through the intermediate steps leading up to the HDP-PCFG—the Bayesian finite mixture model (Appendix A.1.1), the DP mixture model (Appendix A.1.2), and the HDP-HMM (Appendix A.1.3).

2.1 Model definition

Figure 2 defines the generative process for the HDP-PCFG, which consists of two stages: we first generate the grammar (which includes the rule probabilities) specified by (β, ϕ); then we generate a parse tree and its yield (z, x) using that grammar. To generate the grammar, we first draw a countably infinite set of stick-breaking probabilities, $\beta \sim \text{GEM}(\alpha)$, which provides us with a base distribution over grammar symbols, represented by the positive integers (Figure 3(a); see Appendix A.1.2 for a formal definition of the stick-breaking distribution). Next, for each symbol $z = 1, 2, \dots$, we generate the probabilities of the rules of the form $z \to \gamma$, i.e., those which condition on z as the left-hand side. The emission probabilities ϕ_z^E are drawn from a Dirichlet distribution, and provide multinomial distributions over terminal symbols Σ. For the binary production probabilities, we

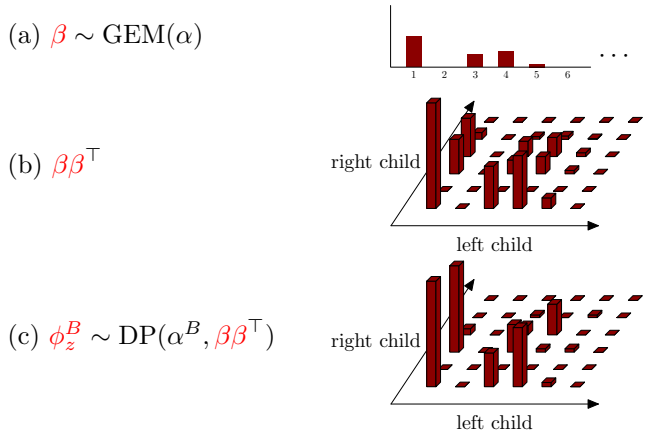


Figure 3: The generation of binary production probabilities given the top-level symbol probabilities β. First, β is drawn from the stick-breaking prior, as in any DP-based model (a). Next, the outer-product $\beta\beta^\top$ is formed, resulting in a doubly-infinite matrix (b). We use this as the base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

first form the product distribution $\beta\beta^\top$ (Figure 3(b)), represented as a doubly-infinite matrix. The binary production probabilities ϕ_z^B are then drawn from a Dirichlet process with base distribution $\beta\beta^\top$—this provides multinomial distributions over pairs of nonterminal symbols (Figure 3(c)). The Bayesian hierarchy ties these distributions together through the base distribution over symbols, so that the grammar effectively has a globally shared inventory of symbols. Note that the HDP-PCFG hierarchy ties distributions over symbol pairs via distributions over single symbols, in contrast to the hierarchy in the standard HDP-HMM, where the distributions being tied are defined over the same space as that of the base distribution. Finally, we generate a "switching" distribution ϕ_z^T over the two rule types $\{\textsc{Emission}, \textsc{Binary-Production}\}$ from a symmetric Dirichlet. The shapes of all the Dirichlet distributions and the Dirichlet processes in our models are governed by concentration hyperparameters: $\alpha, \alpha^T, \alpha^B, \alpha^E$.

Given a grammar, we generate a parse tree and sentence in the same manner as for an ordinary PCFG: start with the root node having the designated root symbol. For each nonterminal node i, we first choose a rule type t_i using $\phi_{z_i}^T$ and then choose a rule with that type—either an emission from $\mathrm{Mult}(\phi_{z_i}^E)$ producing x_i or a binary production from $\mathrm{Mult}(\phi_{z_i}^T)$ producing $\left(z_{c_1(i)}, z_{c_2(i)}\right)$. We recursively apply the procedure on any nonterminal children.

2.2 Partitioning the preterminal and constituent symbols

It is common to partition nonterminal grammar symbols into those that emit only terminals (*preterminal symbols*) and those that produce only two child grammar symbols (*constituent symbols*). An easy way to accomplish this is to let $\alpha^T = (0, 0)$. The resulting $\mathrm{Dir}(0, 0)$ prior on the rule type probabilities forces any draw $\phi_z^T \sim \mathrm{Dir}(\alpha^T)$ to put mass on only one rule type. Despite the simplicity of this approach, is not suitable for our inference method, so let us make the partitioning more explicit.

Define two disjoint inventories, one for preterminal symbols ($\beta^P \sim \mathrm{GEM}(\alpha)$) and one for constituent symbols ($\beta^C \sim \mathrm{GEM}(\alpha)$). Then, we can define the following four rule types: preterminal-preterminal productions with rule probabilities $\phi_z^{PP} \sim \mathrm{DP}\left(\alpha^{PP}, \beta^P(\beta^P)^\top\right)$, preterminal-constituent productions with $\phi_z^{PC} \sim \mathrm{DP}\left(\alpha^{PC}, \beta^P(\beta^C)^\top\right)$, constituent-preterminal productions with $\phi_z^{CP} \sim \mathrm{DP}\left(\alpha^{CP}, \beta^C(\beta^P)^\top\right)$, and constituent-constituent productions with $\phi_z^{CC} \sim \mathrm{DP}\left(\alpha^{CC}, \beta^C(\beta^C)^\top\right)$. Each node has a symbol which is either a preterminal (P, z) or a constituent (C, z). In the former case, a terminal is produced from ϕ_z^E. In the latter case, one of the four rule types is first

chosen given ϕ_z^T and then a rule of that type is chosen.

2.3 Other nonparametric grammars

An alternative definition of an HDP-PCFG would be as follows: for each symbol z, draw a distribution over left child symbols $\phi_z^{B_1} \sim \text{DP}(\alpha, \beta)$ and an independent distribution over right child symbols $\phi_z^{B_2} \sim \text{DP}(\alpha, \beta)$. Then define the binary production distribution as the product $\phi_z^B = \phi_z^{B_1} \phi_z^{B_2 \top}$. This also yields a distribution over symbol pairs and defines a different type of nonparametric PCFG. This model is simpler than the HDP-PCFG 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.

Other alternatives to the HDP-PCFG include the *adaptor grammar* framework of Johnson et al. (2006) and the *infinite tree* framework of Finkel et al. (2007). Both of these nonparametric Bayesian models have been developed using the Chinese restaurant process rather than the stick-breaking representation. In an adaptor grammar, each symbol is associated with a distribution over subtrees rather than a distribution over pairs of child symbols as in the PCFG. While this gives adaptor grammars the ability to capture more global properties of a sentence, it also comes with a serious limitation: Adaptor grammars lack recursion (that is, they forbid the coexistence of rules of the form $A \rightarrow BC$ and $B \rightarrow DA$). As a result, applications of adaptor grammars have focused on the modeling of word segmentation and collocation rather than full constituency syntax. The infinite tree framework is more closely related to the HDP-PCFG, differing principally in that it has been developed for dependency parsing rather than constituency parsing.

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

While the HDP-PCFG is suitable for grammar induction, where we try to infer a full grammar from raw sentences, in practice we often have access to *treebanks*, which are collections of tens of thousands of sentences, each hand-labeled with a syntactic parse tree. Not only do these observed trees help constrain the model, they also declare the coarse symbol inventories which are assumed by subsequent linguistic processing. As discussed in Section 1, these treebank trees represent only the coarse syntactic structure. In order to obtain accurate parsing performance, we need to learn a more refined grammar.

We introduce an extension of the HDP-PCFG appropriate for grammar refinement (the HDP-PCFG-GR). The HDP-PCFG-GR differs from the basic HDP-PCFG in that the coarse symbols are fixed and only their subsymbols are modeled and controlled via the HDP machinery. This formulation makes explicit the level of coarse, observed symbols and provides implicit control over model complexity within those symbols using a single, unified probabilistic model.

3.1 Model definition

The essential difference in the grammar refinement setting is that now we have a collection of HDP-PCFG models, one for each symbol $s \in S$, and each HDP-PCFG operates at the subsymbol level. In practical applications we also need to allow unary productions—a symbol producing exactly one child symbol—the PCFG counterpart of transitions in HMMs. Finally, each nonterminal node $i \in N$ in the parse tree has a symbol-subsymbol pair (s_i, z_i), so each subsymbol needs to specify a distribution over both child symbols and subsymbols. The former is handled through a finite Dirichlet distribution since the finite set of symbols is known. The latter is handled with the HDP machinery, since the (possibly unbounded) set of subsymbols is unknown.

Despite the apparent complexity of the HDP-PCFG-GR model, it is fundamentally a PCFG where the symbols are (s, z) pairs with $s \in S$ and $z \in \{1, 2, \dots\}$. The rules of this PCFG take one of three forms: $(s, z) \to x$ for some terminal symbol $x \in \Sigma$, unary productions $(s, z) \to (s', z')$ and binary productions $(s, z) \to (s', z')(s'', z'')$. We can think of having a single (infinite) multinomial distribution over all right-hand sides given (s, z). Interestingly, the prior distribution of these rule probabilities is no longer a Dirichlet or a Dirichlet process, but rather a more general Pólya tree over distributions on $\{\Sigma \cup (S \times Z) \cup (S \times Z \times S \times Z)\}$.

Figure 4 provides the complete description of the generative process. As in the HDP-PCFG, we first generate the grammar: For each (coarse) symbol s, we generate a distribution over its subsymbols β_s, and then for each subsymbol z, we generate a distribution over right-hand sides via a set of conditional distributions. Next, we generate a distribution ϕ^T_{sz} over the three rule types $\{\text{EMISSION}, \text{UNARY-PRODUCTION}, \text{BINARY-PRODUCTION}\}$. The emission distribution ϕ^T_{sz} is drawn from a Dirichlet as before. For the binary rules, we generate from a Dirichlet a distribution ϕ^b_{sz} over pairs of coarse symbols and then for each coarse symbol $s' \in S$, we generate from a Dirichlet process a distribution $\phi^b_{szs's''}$ over pairs of subsymbols (z', z''). The unary probabilities are generated analogously. Given the grammar, we generate a parse tree and a sentence via a recursive process similar to the one for the HDP-PCFG; the main difference is that at each node, we first generate the coarse symbols of its children and then the child subsymbols conditioned on the coarse symbols.

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

For each symbol $s \in S$:
- $\beta_s \sim \text{GEM}(\alpha)$ [draw subsymbol probabilities]
- For each subsymbol $z \in \{1, 2, \dots\}$:
  - $\phi^T_{sz} \sim \text{Dir}(\alpha^T)$ [draw rule type parameters]
  - $\phi^E_{sz} \sim \text{Dir}(\alpha^E(s))$ [draw emission parameters]
  - $\phi^u_{sz} \sim \text{Dir}(\alpha^u(s))$ [draw unary symbol production parameters]
  - For each child symbol $s' \in S$:
    - $\phi^U_{szs'} \sim \text{DP}(\alpha^U, \beta_{s'})$ [draw unary subsymbol production parameters]
  - $\phi^b_{sz} \sim \text{Dir}(\alpha^b(s))$ [draw binary symbol production parameters]
  - For each pair of child symbols $(s', s'') \in S \times S$:
    - $\phi^B_{szs's''} \sim \text{DP}(\alpha^B, \beta_{s'}\beta^\top_{s''})$ [draw binary subsymbol production parameters]

For each node i in the parse tree:
- $t_i \sim \text{Mult}(\phi^T_{s_i z_i})$ [choose rule type]
- If $t_i = \text{EMISSION}$:
  - $x_i \sim \text{Mult}(\phi^E_{s_i z_i})$ [emit terminal symbol]
- If $t_i = \text{UNARY-PRODUCTION}$:
  - $s_{c_1(i)} \sim \text{Mult}(\phi^u_{s_i z_i})$ [generate child symbol]
  - $z_{c_1(i)} \sim \text{Mult}(\phi^U_{s_i z_i s_{c_1(i)}})$ [generate child subsymbol]
- If $t_i = \text{BINARY-PRODUCTION}$:
  - $(s_{c_1(i)}, s_{c_2(i)}) \sim \text{Mult}(\phi_{s_i z_i})$ [generate child symbols]
  - $(z_{c_1(i)}, z_{c_2(i)}) \sim \text{Mult}(\phi^B_{s_i z_i s_{c_1(i)} s_{c_2(i)}})$ [generate child subsymbols]

Figure 4: The definition of the HDP-PCFG for grammar refinement (HDP-PCFG-GR).

The HDP-PCFG-GR is a generalization of many of the DP-based models in the literature. When there is exactly one symbol ($|S| = 1$), it reduces to the HDP-PCFG, where the subsymbols in the HDP-PCFG-GR play the role of symbols in the HDP-PCFG. Suppose we have two distinct symbols A and B and that all trees are three node chains of the form $A \to B \to x$. Then the HDP-PCFG-GR is equivalent to the nested Dirichlet process (Rodriguez et al., 2008). If the subsymbols of A are

observed for all data points, we have a plain hierarchical Dirichlet process (Teh et al., 2006), where the subsymbol of A corresponds to the group of data point x.

4 Bayesian inference

In this section, we describe an approximate posterior inference algorithm for the HDP-PCFG based on variational inference. We present an algorithm only for the HDP-PCFG; the extension to the HDP-PCFG-GR is straightforward, requiring only additional bookkeeping. For the basic HDP-PCFG, Section 4.1 describes mean-field approximations of the posterior. Section 4.2 discusses the optimization of this approximation. Finally, Section 4.3 addresses the use of this approximation for predicting parse trees on new sentences. For a further discussion of the variational approach, see Appendix A.2.1. Detailed derivations are presented in Appendix B.

4.1 Structured mean-field approximation

The random variables of interest in our model are the parameters $\theta = (\beta, \phi)$, the parse tree z, and the yield x (which we observe). Our goal is thus to compute the posterior $p(\theta, z \mid x)$. We can express this posterior variationally as the solution to an optimization problem:

$$\operatorname*{argmin}_{q \in \mathcal{Q}} \mathrm{KL}(q(\theta, z) \,||\, p(\theta, z \mid x)). \tag{2}$$

Indeed, if we let $\mathcal{Q}$ be the family of all distributions over (θ, z), the solution to the optimization problem is the exact posterior $p(\theta, z \mid x)$, since KL divergence is minimized exactly when its two arguments are equal. Of course, solving this optimization problem is just as intractable as directly computing the posterior. Though it may appear that no progress has been made, having a variational formulation allows us to consider tractable choices of $\mathcal{Q}$ in order to obtain principled approximate solutions.

For the HDP-PCFG, we define the set of approximate distributions $\mathcal{Q}$ to be those that factor as follows:

$$\mathcal{Q} = \Big\{ q : \Big[q(\beta) \prod_{z=1}^{K} q(\phi_z^T) q(\phi_z^E) q(\phi_z^B) \Big] q(z) \Big\}. \tag{3}$$

Figure 5 shows the graphical model corresponding to the family of approximate distributions we consider. Furthermore, we impose additional parametric forms on the factors as follows:

- $q(\beta)$ is degenerate ($q(\beta) = \delta_{\beta^*}(\beta)$ for some β^*) and truncated ($\beta_z^* = 0$ for $z > K$).

  The truncation is typical of inference algorithms for DP mixtures that are formulated using the stick-breaking representation (Blei and Jordan, 2005). It can be justified by the fact that a truncated stick-breaking distribution well approximates the original stick-breaking distribution in the following sense: Let $G = \sum_{z=1}^{\infty} \beta_z \delta_{\phi_z}$ and let $G^K = \sum_{z=1}^{K} \beta'_z \delta_{\phi_z}$ denote the truncated version, where $\beta'_z = \beta_z$ for $z < K$, $\beta'_K = 1 - \sum_{z=1}^{K-1} \beta_z$ and $\beta'_z = 0$ for $z > K$. The variational distance between G and G^K decreases exponentially as a function of the truncation level K (Ishwaran and James, 2001).

  We also require $q(\beta)$ to be degenerate to avoid the computational difficulties due to the nonconjugacy of β and ϕ_z^B.

- $q(\phi_z^T)$, $q(\phi_z^E)$, and $q(\phi_z^B)$ are Dirichlet distributions. Although the binary production parameters ϕ_z^B specify a distribution over $\{1, 2, \dots\}^2$, the K-component truncation on β forces ϕ_z^B to assign zero probability to all but a finite $K \times K$ subset. Therefore, only a finite Dirichlet distribution and not a general DP is needed to characterize the distribution over ϕ_z^B.

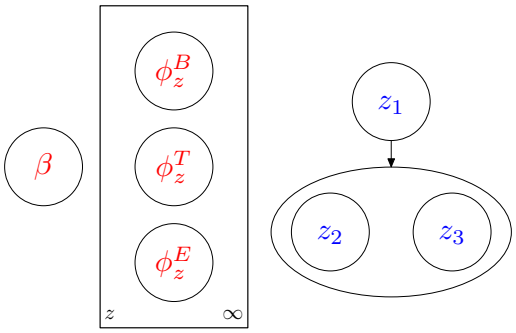


Figure 5: We approximate the posterior over parameters and parse trees using structured mean-field. Note that the posterior over parameters is completely factorized, but the posterior over parse trees is not constrained.

- $q(z)$ is any multinomial distribution (this encompasses all possible distributions over the discrete space of parse trees). Though the number of parse trees is exponential in the sentence length, it will turn out that the optimal $q(z)$ always has a factored form which can be found efficiently using dynamic programming.

Note that if we had restricted all the parameter distributions to be degenerate ($q(\phi) = \delta_{\phi^*}(\phi)$ for some ϕ^*) and fixed β^* to be uniform, then we would get the objective function optimized by the EM algorithm. If we further restrict $q(z) = \delta_{z^*}(z)$ for some z^*, we would obtain the objective optimized by Viterbi EM.

4.2 Coordinate ascent

We now present an algorithm for solving the optimization problem described in (2) and (3). Unfortunately, the optimization problem is non-convex, and it is intractable to find the global optimum. However, we can use a simple coordinate ascent algorithm to find a local optimum. The algorithm optimizes one factor in the mean-field approximation of the posterior at a time while fixing all other factors. Optimizing $q(z)$ is the analog of the E-step in fitting an ordinary PCFG, and optimizing $q(\phi)$ is the analog of the M-step. Optimizing $q(\beta)$ has no analog in EM. Note that Kurihara and Sato (2004) proposed a structured mean-field algorithm for Bayesian PCFGs; like ours, their algorithm optimizes $q(z)$ and $q(\phi)$, but it does not optimize $q(\beta)$.

Although mathematically our algorithm performs coordinate ascent on the mean-field factors shown in (3), the actual implementation of the algorithm does not actually require storing each factor explicitly; it stores only summaries sufficient for updating the other factors. For example, to update $q(\phi)$, we only need the expected counts derived from $q(z)$; to update $q(z)$, we only need the multinomial weights derived from $q(\phi)$ (see below for details).

4.2.1 Updating posterior over parse trees $q(z)$ ("E-step")

A sentence is a sequence of terminals $x = (x_1, \ldots, x_n)$; conditioned on the sentence length n, a parse tree can represented by the following set of variables: $z = \{z_{[i,j]} : 1 \leq i \leq j \leq n\}$, where the variable $z_{[i,j]}$ indicates whether there is a node in the parse tree whose terminal span is $(x_i, \ldots, x_j)$, and, if so, what the grammar symbol at that node is. In the first case, the value of $z_{[i,j]}$ is a grammar symbol, in which case we call $[i, j]$ a *constituent*; otherwise, $z_{[i,j]}$ takes on a special NON-NODE value. In order for z to specify a valid parse tree, two conditions must hold: (1) $[1, n]$ is a constituent, and (2) for each constituent $[i, j]$ with $i < j$, there exists exactly one $k \in [i, j-1]$ for which $[i, k]$ and $[k+1, j]$ are constituents.

Before deriving the update for $q(z)$, we introduce some notation. Let $B(z) = \{[i,j] : i < j \text{ and } z_{[i,j]} \neq \text{NON-NODE}\}$ be the set of binary constituents. Let $c_1([i,j]) = [i,k]$ and $c_2([i,j]) = [k+1,j]$, where k is the unique integer such that $[i,k]$ and $[k+1,j]$ are constituents.

The conditional distribution over the parse tree is given by the following:

$$\begin{aligned} p(z \mid \theta, x) &\propto p(x, z \mid \theta) \\ &= \prod_{i=1}^{n} \phi^T_{z[i,i]}(\text{E})\, \phi^E_{z_{[i,i]}}(x_i) \prod_{[i,j] \in B(z)} \phi^T_{z[i,j]}(\text{B})\, \phi^B_{z_{[i,j]}}(z_{c_1([i,j])} z_{c_2([i,j])}), \end{aligned} \tag{4}$$

where we have abbreviated the rule types: E = EMISSION and B = BINARY-PRODUCTION. The first product is over the emission probabilities of generating each terminal symbol x_i, and the second product is over the binary production probabilities used to produce the parse tree. Using this conditional distribution, we can form the mean-field update by applying the general mean-field update rule (see (35) in Appendix B.1):

$$q(z) \propto \prod_{i=1}^{n} W^T_{z_{[i,i]}}(\text{E})\, W^E_{z_{[i,i]}}(x_i) \prod_{[i,j] \in B(z)} W^T_{z_{[i,j]}}(\text{B})\, W^B_{z_{[i,j]}}(z_{c_1([i,j])}, z_{c_2([i,j])}), \tag{5}$$

where the multinomial weights are defined as follows:

$$W^E_z(x) \stackrel{\text{def}}{=} \exp\{E_{q(\phi)} \log \phi^E_z(x)\}, \tag{6}$$

$$W^B_z(z', z'') \stackrel{\text{def}}{=} \exp\{E_{q(\phi)} \log \phi^B_z(z'z'')\}, \tag{7}$$

$$W^T_z(t) \stackrel{\text{def}}{=} \exp\{E_{q(\phi)} \log \phi^T_z(t)\}. \tag{8}$$

These multinomial weights for a d-dimensional Dirichlet are simply d numbers that "summarize" that distribution. Note that if $q(\phi)$ were degenerate, then W^T, W^E, W^B would be equal to their non-random counterparts ϕ^T, ϕ^E, ϕ^B. In this case, the mean-field update (5) is the same as conditional distribution (4). Even when $q(\phi)$ is not degenerate, note that (4) and (5) factor in the same way, which has important implications for computational efficiency.

To compute the normalization constant of (5), we can use using dynamic programming. In the context of PCFGs, this amounts to using the standard inside-outside algorithm (Lari and Young, 1990); see Manning and Schütze (1999) for details. This algorithm runs in $O(n^3K^3)$ time, where n is the length of the sentence and K is the truncation level. However, if the tree structure is fixed as in the case of grammar refinement, then we can use a variant of the forward-backward algorithm which runs in just $O(nK^3)$ time (Matsuzaki et al., 2005; Petrov et al., 2006).

A common perception is 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 imported into the Bayesian framework in a modular way. In Section A.3.2, we give an empirical interpretation of the multinomial weights, showing how they take into account the uncertainty of the random rule probabilities they represent.

4.2.2 Updating posterior over rule probabilities $q(\phi)$ ("M-step")

The mean-field update for $q(\phi)$ can be broken up into independent updates for the rule type parameters $q(\phi^T_z)$ the emission parameters $q(\phi^E_z)$ and the binary production parameters $q(\phi^B_z)$ for each symbol $z \in S$.

Just as optimizing $q(z)$ only required the multinomial weights of $q(\phi)$, optimizing $q(\phi)$ only requires a "summary" of $q(z)$. In particular, this summary consists of the expected counts of emissions,

rule types, and binary productions, which can be computed by the inside-outside algorithm:

$$C^E(z,x) \stackrel{\text{def}}{=} E_{q(z)} \sum_{1\le i\le n} \mathbb{I}[z_{[i,i]} = z, x_i = x], \tag{9}$$

$$C^B(z,z'\,z'') \stackrel{\text{def}}{=} E_{q(z)} \sum_{1\le i\le k<j\le n} \mathbb{I}[z_{[i,j]} = z, z_{[i,k]} = z', z_{[k+1,j]} = z''], \tag{10}$$

$$C^T(z,t) \stackrel{\text{def}}{=} \sum_{\gamma} C^t(z,\gamma). \tag{11}$$

Applying (35), we obtain the following updates (see Appendix B.3 for a derivation):

$$\begin{aligned} q(\phi_z^E) &= \text{Dir}(\phi_z^E; \alpha^E + C^E(z)), \\ q(\phi_z^B) &= \text{Dir}(\phi_z^B; \alpha^B + C^B(z)), \\ q(\phi_z^T) &= \text{Dir}(\phi_z^T; \alpha^T + C^T(z)). \end{aligned}$$

4.2.3 Updating the top-level component weights $q(\beta)$

Finally, we need to update the parameters in the top level of our Bayesian hierarchy $q(\beta) = \delta_{\beta^*}(\beta)$. Unlike the other updates, there is no closed-form expression for the optimal β^*. In fact, the objective function (2) is not even convex in β^*. Nonetheless, we can use a projected gradient algorithm (Bertsekas, 1999) to improve β^* to a local optimum. The optimization problem is as follows:

$$\min_{\beta^*} \text{KL}\left(q(\theta,z) \,||\, p(\theta,z \mid x)\right) \tag{12}$$

$$= \max_{\beta^*} E_q \log p(\theta,z \mid x) + H(q(\theta,z)) \tag{13}$$

$$= \max_{\beta^*} \log \text{GEM}(\beta^*;\alpha) + \sum_{z=1}^{K} E_q \log \text{Dir}(\phi_z^B; \alpha^B \beta^* \beta^{*T}) + \text{constant} \tag{14}$$

$$\stackrel{\text{def}}{=} \max_{\beta^*} L(\beta^*) + \text{constant}. \tag{15}$$

We have absorbed all terms that do not depend on β^* into the constant, including the entropy, since δ_{β^*} is always degenerate. See Appendix B.4 for the details of the gradient projection algorithm and the derivation of $L(\beta)$ and $\nabla L(\beta)$.

4.3 Prediction: parsing new sentences

After having found an approximate posterior over parameters of the HDP-PCFG, we would like to be able to use it to parse new sentences, that is, predict their parse trees. Given a loss function $\ell(z,z')$ between parse trees, the Bayes optimal parse tree for a new sentence x_{new} is given as follows:

$$z^*_{\text{new}} = \operatorname*{argmin}_{z'_{\text{new}}} E_{p(z_{\text{new}} \mid x_{\text{new}}, x, z)} \ell(z_{\text{new}}, z'_{\text{new}}) \tag{16}$$

$$= \operatorname*{argmin}_{z'_{\text{new}}} E_{p(z_{\text{new}} \mid \theta, x_{\text{new}}) p(\theta, z \mid x)} \ell(z_{\text{new}}, z'_{\text{new}}). \tag{17}$$

If we use the 0-1 loss $\ell(z_{\text{new}}, z'_{\text{new}}) = 1 - \mathbb{I}[z_{\text{new}} = z'_{\text{new}}]$, then (16) is equivalent to finding the maximum marginal likelihood parse, integrating out the parameters θ:

$$z^*_{\text{new}} = \operatorname*{argmax}_{z_{\text{new}}} E_{p(\theta,z \mid x)} p(z_{\text{new}} \mid \theta, x_{\text{new}}). \tag{18}$$

We can substitute in place of the true posterior $p(\theta, z \mid x)$ our approximate posterior $q(\theta, z) = q(\theta)q(z)$ to get an approximate solution:

$$z^*_{\text{new}} \quad = \quad \operatorname*{argmax}_{z_{\text{new}}} E_{q(\theta)} p(z_{\text{new}} \mid \theta, x_{\text{new}}). \tag{19}$$

If $q(\theta)$ were degenerate, then we could evaluate the argmax efficiently using dynamic programming (the Viterbi version of the inside algorithm). However, even for a fully-factorized $q(\theta)$, the argmax cannot be computed efficiently, as noted by MacKay (1997) in the context of variational HMMs. The reason behind this difficulty is that the integration over that rule probabilities couples distant parts of the parse tree which use the same rule, thus destroying the Markov property necessary for dynamic programming. We are thus left with the following options:

1. Maximize the expected log probability instead of the expected probability:

$$z^*_{\text{new}} \quad = \quad \operatorname*{argmax}_{z_{\text{new}}} E_{q(\theta)} \log p(z_{\text{new}} \mid x_{\text{new}}, \theta). \tag{20}$$

   Concretely, this amounts to running the Viterbi algorithm with the same multinomial weights that were used while fitting the HDP-PCFG.

2. Extract the mode $\theta^* = q(\theta)$ and parse using θ^* with dynamic programming. One problem with this approach is that the mode is not always well defined, for example, when the Dirichlet posteriors have concentration parameters less than 1.

3. Use options 1 or 2 to obtain a list of good candidates, and then choose the best candidate according to the true objective (19).

In practice, we used the first option.

4.3.1 Grammar refinement

Given a new sentence x_{new}, recall that the HDP-PCFG-GR (Section 3.1) defines a joint distribution over a coarse tree s_{new} of symbols and a refinement z_{new} described by subsymbols. Finding the best refined tree $(s_{\text{new}}, z_{\text{new}})$ can be carried out using the same methods as for the HDP-PCFG described above. However, in practical parsing applications, we are interested in predicting coarse trees for use in subsequent processing. For example, we may wish to minimize the 0-1 loss with respect to the coarse tree $(1 - \mathbb{I}[s_{\text{new}} = s'_{\text{new}}])$. In this case, we need to integrate out z_{new}:

$$s^*_{\text{new}} \quad = \quad \operatorname*{argmax}_{s_{\text{new}}} E_{q(\theta)} \sum_{z_{\text{new}}} p(s_{\text{new}}, z_{\text{new}} \mid \theta, x_{\text{new}}). \tag{21}$$

Note that even if $q(\theta)$ is degenerate, this expression is difficult to compute because the sum over z_{new} induces long-range dependencies (which is the whole point of grammar refinement), and as a result, $p(s_{\text{new}} \mid x_{\text{new}}, \theta)$ does not decompose and cannot be handled via dynamic programming.

Instead, we adopt the following two-stage strategy, which works quite well in practice (cf. Matsuzaki et al., 2005). We first construct an approximate distribution $\tilde{p}(s \mid x)$ which does decompose and then find the best coarse tree with respect to $\tilde{p}(s \mid x)$ using the Viterbi algorithm:

$$s^*_{\text{new}} = \operatorname*{argmax}_{s} \tilde{p}(s \mid x). \tag{22}$$

We consider only tractable distributions $\tilde{p}$ which permit dynamic programming. These distributions can be interpreted as PCFGs where the new symbols $S \times \{(i, j) : 1 \le i \le j \le n\}$ are annotated with the span. The new rule probabilities must be consistent with the spans; that is, the only rules allowed to have nonzero probability are of the form $(a, [i, j]) \to (b, [i, k])\,(c, [k + 1, j])$ (binary) and

$(a, [i, j]) \rightarrow (b, [i, j])$ (unary), where $a, b, c \in S$ and $x \in \Sigma$. The "best" such distribution is found according to a KL-projection:

$$\tilde{p}(s \mid x) = \underset{\tilde{p}' \text{ tractable}}{\operatorname{argmin}} \ \mathrm{KL}\left(\exp E_{q(\theta)} \log p(s_{\text{new}} \mid x_{\text{new}}, \theta) \,||\, \tilde{p}'(s \mid x)\right), \tag{23}$$

This KL-projection can be done by simple moment-matching, where the moments we need to compute are of the form $\mathbb{I}[(s_{[i,j]} = a, s_{[i,k]} = b, s_{[k+1,j]} = c)]$ and $\mathbb{I}[s_{[i,j]} = a, s_{[i,j]} = b]$ for $a, b, c \in S$, which can be computed using the dynamic programming algorithm described in Section 4.2.1.

Petrov and Klein (2007) discuss several other alternatives, but show that the two-step strategy described above performs the best.

5 Experiments

We now present an empirical evaluation of the HDP-PCFG and HDP-PCFG-GR models for grammar induction and grammar refinement, respectively. We first show that the HDP-PCFG and HDP-PCFG-GR are able to recover a known grammar more accurately than a standard PCFG estimated with maximum likelihood (Sections 5.1 and 5.2). We then present results on a large-scale parsing task (Section 5.3).

5.1 Inducing a synthetic grammar using the HDP-PCFG

In the first experiment, the goal was to recover a PCFG grammar given only sentences generated from that grammar. Consider the leftmost grammar in Figure 6; this is the "true grammar" that we wish to recover. It has a total of eight nonterminal symbols, four of which are left-hand sides of only emission rules (these are the preterminal symbols) and four of which are left-hand sides of only binary production rules (constituent symbols). The probability of each rule is given above the appropriate arrow. Though very simple, this grammar still captures some of the basic phenomena of natural language such as noun phrases (NPs), verb phrases (VPs), determiners (DTs), adjectives (JJs), and so on.

From this grammar, we sampled 1000 sentences. Then, from these sentences alone, we attempted to recover the grammar using the standard PCFG and the HDP-PCFG. For the standard PCFG, we allocated 20 latent symbols, 10 for preterminal symbols and 10 for constituent symbols. For the HDP-PCFG (using the version described in Section 2.2), we set the stick-breaking truncation level to $K = 10$ for the preterminal and constituent symbols. All Dirichlet hyperparameters were set to 0.01. We ran 300 iterations of EM for the standard PCFG and variational inference for the HDP-PCFG. Since both algorithms are prone to local optima, we ran the algorithms 30 times with different random initializations. We also encouraged the use of constituent symbols by placing slightly more prior weight on rule type probabilities corresponding to production-production rules.

In none of the 30 trials did the standard PCFG manage to recover the true grammar. We say a rule is *active* if its multinomial weight (see Section 4.2.1) is at least 10^{-6} and its left-hand side has total posterior probability also at least 10^{-6}. In general, rules with weight smaller than 10^{-6} can be safely ignored without affect parsing results. In a typical run of the standard PCFG, all 20 symbols were used and about 150 of the rules were active (in contrast, there are only 15 rules in the true grammar). The HDP-PCFG managed to do much better. Figure 6 shows three grammars (of the 30 trials) which had the highest variational objective values. The symbols are numbered 1 to K in the model, but we have manually labeled them with to suggest their actual role in the syntax. Each grammar has around 4–5 constituent symbols (there were 4 in the true grammar) and 6–7 preterminal symbols (4 in the true grammar), which yields around 25 active rules (15 in the true grammar).

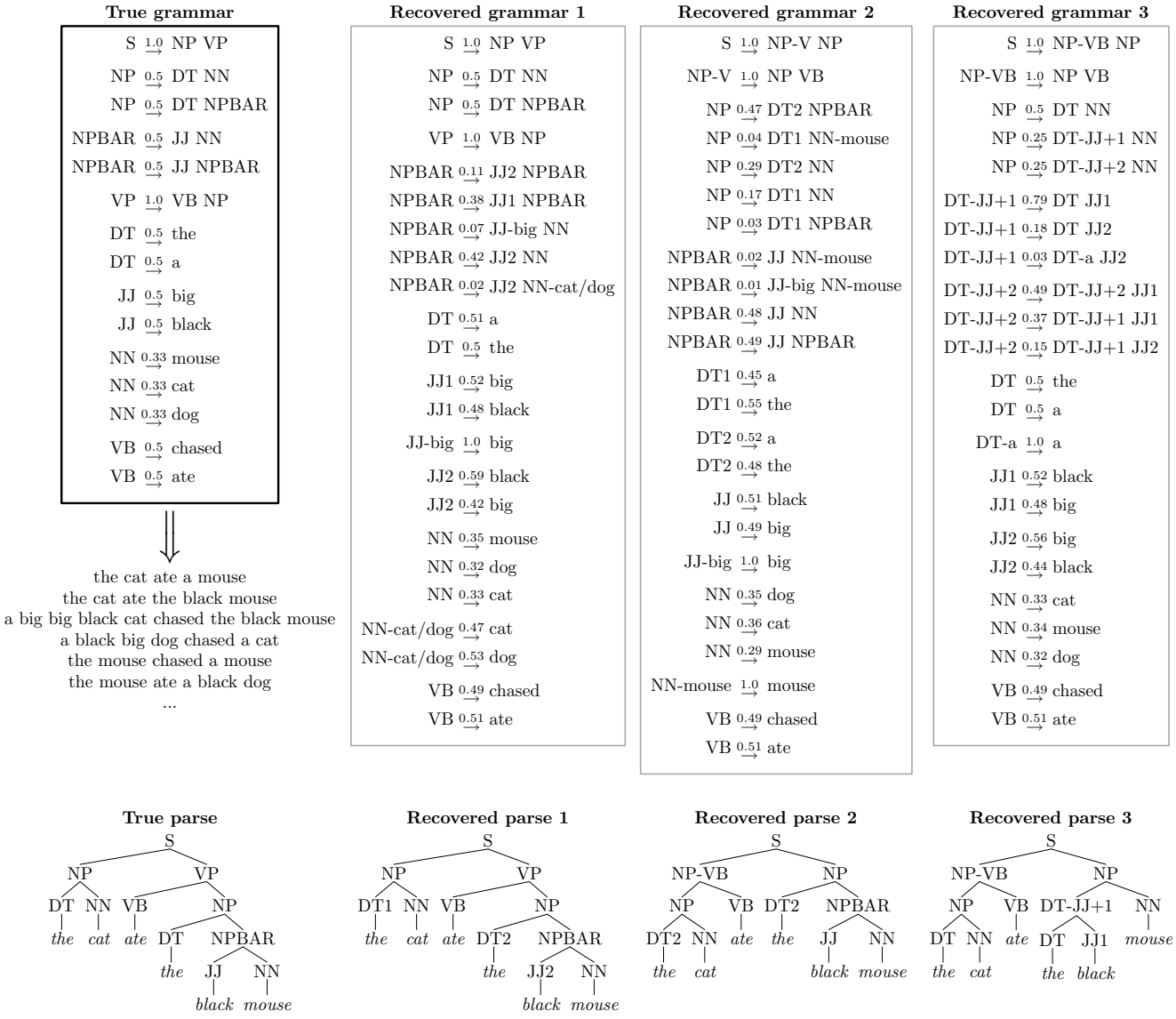


Figure 6: We generated 1000 sentences from the true grammar. The HDP-PCFG model recovered the three grammars (obtained from different initializations of the variational inference algorithm). The parse trees of a sentence under the various grammars are also shown. The first is essentially the same as that of the true grammar, while the second and third contain various left-branching alternatives.

While the HDP-PCFG did not recover the exact grammar, it was able to produce grammars that were sensible. Interestingly, each of the three grammars used a slightly different type of syntactic structure to model the data. The first one is the closest to the true grammar and differs only in that it allocated three subsymbols for JJ and two for NN instead of one. Generally, extra subsymbols provide a better fit of the noisy observations, and here, the sparsity-inducing DP prior was only partially successful in producing a parsimonious model.

The second grammar differs fundamentally from the true grammar in that the subject and the verb are grouped together as a constituent (NP-VB) rather than the verb and the object (VP). This is a problem with non-identifiability; in this simple example, both grammars describe the data equally well and have the same grammar complexity. One way to break this symmetry is to use an informative prior—e.g., that natural language structures tend to be right-branching.

The third grammar differs from the true grammar in one additional way, namely that noun phrases are also left-branching; that is, for *the black mouse*, *the* and *black* are grouped together rather than *black* and *mouse*. Intuitively, this grammar suggests the determiner as the head word of the phrase rather than the noun. This head choice is not entirely unreasonable; indeed there is an ongoing debate in the linguistics community about whether the determiner (the DP hypothesis) or the noun (the NP hypothesis) should be the head (though even in a DP analysis the determiner and adjective should not be grouped).

Given that our variational algorithm converged to various modes depending on initialization, it is evident that the true Bayesian posterior over grammars is multimodal. A fully Bayesian inference procedure would explore and accurately represent these modes, but this is computationally intractable. Starting our variational algorithm from various initializations provides a cheap, if imperfect, way of exploring some of the modes.

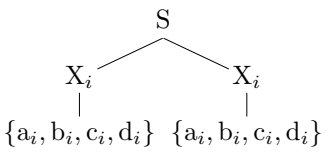
Inducing grammars from raw text alone is an extremely difficult problem. Statistical approaches to grammar induction have been studied since the early 1990s (Carroll and Charniak, 1992). As these early experiments were discouraging, people turned to alternative algorithms (Stolcke and Omohundro, 1994) and models (Klein and Manning, 2004; Smith and Eisner, 2005). Though we have demonstrated partial success with the HDP-PCFG on synthetic examples, it is unlikely that this method alone will solve grammar induction problems for large-scale corpora. Thus, for the remainder of this section, we turn to the more practical problem of grammar refinement, where the learning of symbols is constrained by a coarse treebank.

5.2 Refining a synthetic grammar with the HDP-PCFG-GR

We first conduct a simple experiment, similar in spirit to the grammar induction experiment, to show that the HDP-PCFG-GR can recover a simple grammar while a standard PCFG-GR (a PCFG adapted for grammar refinement) cannot. From the grammar in Figure 7(a), we generated 2000 trees of the form shown in Figure 7(b). We then replaced all X_is with X in the training data to yield a coarse grammar. Note that the two terminal symbols always have the same subscript, a correlation not captured by their parent X. We trained both the standard PCFG-GR and the HDP-PCFG-GR using the modified trees as the input data, hoping to estimate the proper refinement of X. We used a truncation of $K = 20$ for both S and X, set all hyperparameters to 1, and ran EM and the variational algorithm for 100 iterations.

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

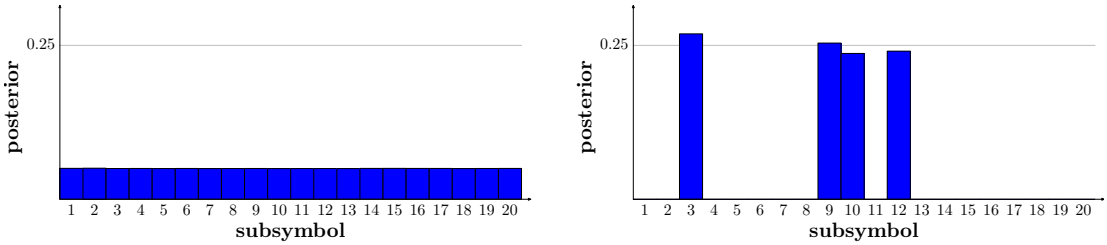
(a)



(b)

Figure 7: (a) A synthetic grammar with a uniform distribution over rules. (b) The grammar generates trees of the form shown on the right.

Figure 8 shows the posterior probabilities of the subsymbols in the grammars produced by the PCFG-GR (a) and HDP-PCFG-GR (b). The PCFG-GR used all 20 subsymbols of both S and X to fit the noisy co-occurrence statistics of left and right terminals, resulting in 8320 active rules (with multinomial weight larger than 10^{-6}). On the other hand, for the HDP-PCFG-GR, only four subsymbols of X and one subsymbol of S had non-negligible posterior mass; 68 rules were active. If the threshold is relaxed from 10^{-6} to 10^{-3}, then only 20 rules are active, which corresponds exactly to the true grammar.



(a) PCFG-GR (b) HDP-PCFG-GR

Figure 8: The posterior probabilities over the subsymbols of the grammar produced by the PCFG-GR are roughly uniform, whereas the posteriors for the HDP-PCFG-GR are concentrated on four subsymbols, the true number in the original grammar.

5.3 Parsing the Penn Treebank

In this section, we show that the variational HDP-PCFG-GR can scale up to real-world datasets. We truncated the HDP-PCFG-GR at K subsymbols, and compared its performance with a standard PCFG-GR with K subsymbols estimated using maximum likelihood (Matsuzaki et al., 2005).

5.3.1 Dataset and preprocessing

We ran experiments on the Wall Street Journal (WSJ) portion of the Penn Treebank, a standard dataset used in the natural language processing community for evaluating constituency parsers. The dataset is divided into 24 sections and consists of approximately 40,000 sentences. As is standard, we fit a model on sections 2–21, used section 24 for tuning the hyperparameters of our model, and evaluated parsing performance on section 22.

The HDP-PCFG-GR is defined only for grammars with unary and binary production rules, but the Penn Treebank contains trees in which a node has more than two children. Therefore, we use a standard *binarization* procedure to transform our trees. Specifically, for each nonterminal node with symbol X, we introduce a right-branching cascade of new nodes with symbol $\overline{X}$.

Another issue that arises when dealing with a large-scale dataset of this kind is that words have a Zipfian distribution and in parsing new sentences we invariably encounter new words that did not appear in the training data. We could let the HDP-PCFG-GR's generic Dirichlet prior manage this uncertainty, but would like to use prior knowledge such as the fact that a capitalized word that we have not seen before is most likely a proper noun. We use a simple method to inject this prior knowledge: replace any word appearing fewer than 5 times in the training set with one of 50 special "unknown word" tokens, which are added to Σ. These tokens can be thought of as representing manually constructed word clusters.

We evaluated our predicted parse trees using F_1 score, defined as follows. Given a (coarse) parse tree s, let the labeled brackets be defined as

$$\mathrm{LB}(s) = \{(s_{[i,j]}, [i,j]) : s_{[i,j]} \neq \textsc{Non-Node}, 1 \leq i \leq j \leq n\}.$$

Given the correct parse tree s and a predicted parse tree s', the precision, recall and F_1 scores are defined as follows:

$$\mathrm{Precision}(s, s') = \frac{|\mathrm{LB}(s) \cap \mathrm{LB}(s')|}{|\mathrm{LB}(s)|} \quad \mathrm{Recall}(s, s') = \frac{|\mathrm{LB}(s) \cap \mathrm{LB}(s')|}{|\mathrm{LB}(s')|} \tag{24}$$

$$\mathrm{F}_1(s, s')^{-1} = \frac{1}{2}\left(\mathrm{Precision}(s, s')^{-1} + \mathrm{Recall}(s, s')^{-1}\right). \tag{25}$$

5.3.2 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 greater accuracy 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 α^B is too small, the grammars are overly sparse, but if α^B is too large, the extra smoothing makes the grammars too uniform (see Figure 9).

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.

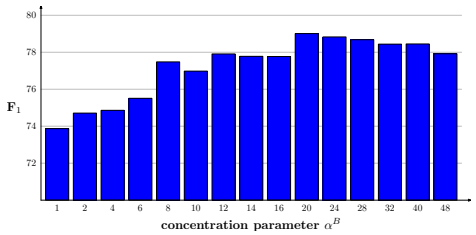


Figure 9: Development F_1 performance on the development set for various values of α^B, training on only section 2 with truncation $K = 8$.

If the top-level distribution β is uniform, the value of α^B corresponding to a uniform prior over pairs of child subsymbols is K^2. Interestingly, the best α^B appears to be superlinear but subquadratic in K. We used these best values of α^B in the following experiments.

5.3.3 Results

The regime in which Bayesian inference is most important is when the number of training data points is small relative to the complexity of the model. To explore this regime, we conducted a first experiment where we trained the PCFG-GR and HDP-PCFG-GR on only section 2 of the Penn Treebank.

Table 2 shows the results of this experiment. Without smoothing, the PCFG-GR trained using EM improves as K increases but starts to overfit around $K = 4$. If we smooth the PCFG-GR by adding 0.01 pseudocounts (which corresponds using a Dirichlet prior with concentration parameters 1.01, ..., 1.01), we see that the performance does not degrade even as K increases to 20, but the number of active grammar rules needed is substantially more. Finally, we see that the HDP-PCFG-GR yields performance comparable to the smoothed PCFG-GR, but the number of grammar rules needed is smaller.

We also conducted a second experiment to demonstrate that our methods can scale up to realistically large corpora. In particular, in this experiment we trained on all of sections 2–21. When using a truncation level of $K = 16$, the standard PCFG-GR with smoothing yielded an F_1 score of 88.36 using 706,157 active rules. The HDP-PCFG-GR yielded an F_1 score of 87.08 using 428,375

K	PCFG-GR		PCFG-GR (smoothed)		HDP-PCFG-GR	
	F_1	Size	F_1	Size	F_1	Size
1	60.47	2558	60.36	2597	60.50	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	78.80	160403	78.94	86386
16	66.99	19616	79.20	261444	78.24	131377
20	64.44	27593	79.27	369699	77.81	202767

Table 2: Shows the development F_1 scores and grammar sizes (the number of active rules) as we increase the truncation K. The ordinary PCFG-GR overfits around $K = 4$. Smoothing with 0.01 pseudocounts prevents this, but produces much larger grammars. The HDP-PCFG-GR attains comparable performance with smaller grammars.

active rules. We thus see that the HDP-PCFG-GR achieves broadly comparable performance compared to existing state-of-the-art parsers, while requiring a substantially smaller number of rules. Finally, note that $K = 16$ is a relatively stringent truncation and we would expect to see improved performance from the HDP-PCFG-GR with a larger truncation level.

6 Discussion

The HDP-PCFG represents a marriage of grammars and Bayesian nonparametrics. We view this marriage as a particularly natural one, given the need for syntactic models to have large, open-ended numbers of grammar symbols. Moreover, given that the most successful methods that have been developed for grammar refinement have been based on clustering of grammar symbols, we view the Dirichlet process as providing a natural starting place for approaching this problem from a Bayesian point of view. The main problem that we have had to face has been that of tying of clusters across parse trees, and this problem is readily solved via the hierarchical Dirichlet process.

We have also presented an efficient variational inference algorithm to approximate the Bayesian posterior over grammars. Although more work is needed to demonstrate the capabilities and limitations of the variational approach, it is important to emphasize the need for fast inference algorithms in the domain of natural language processing. A slow parser is unlikely to make inroads in the applied NLP community. Note also that the mean-field variational algorithm that we presented is closely linked to the EM algorithm, and the familiarity of the latter in the NLP community may help engender interest in the Bayesian approach.

The NLP community is constantly exploring new problem domains that provide fodder for Bayesian modeling. Currently, one very active direction of research is the modeling of multilingual data—for example, synchronous grammars which jointly model the parse trees of pairs of sentences which are translations of each other (Wu, 1997). These models have also received a initial Bayesian treatment (Blunsom et al., 2009). Also, many of the models used in machine translation are essentially already nonparametric, as the parameters of these models are defined on large subtrees rather than individual rules as in the case of the PCFG. Here, having a Bayesian model that integrates over uncertainty can provide more accurate results (DeNero et al., 2008). A major open challenge for these models is again computational; Bayesian inference must be fast if its application to NLP is likely to succeed.

Bayesian finite mixture model

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

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

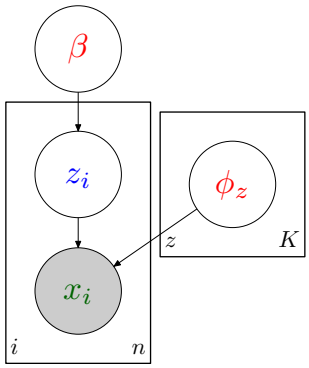


Figure 10: The definition and graphical model of the Bayesian finite mixture model.

A Broader Context and Background

In this section, we present some of the background in nonparametric Bayesian modeling and inference needed for understanding the HDP-PCFG. Appendix A.1 describes various simpler models that lead up to the HDP-PCFG. Appendix A.2 overviews some general issues in variational inference and Appendix A.3 provides a more detailed discussion of the properties of mean-field variational inference in the nonparametric setting.

A.1 DP-based models

We first review the Dirichlet process (DP) mixture model (Antoniak, 1974), a building block for a wide variety of nonparametric Bayesian models, including the HDP-PCFG. The DP mixture model captures the basic notion of clustering which underlies symbol formation in the HDP-PCFG. We then consider the generalization of mixture models to hidden Markov models (HMMs), where clusters are linked structurally according to a Markov chain. To turn the HMM into a nonparametric Bayesian model, we introduce the hierarchical Dirichlet process (HDP). Combining the HDP with the Markov chain structure yields the HDP-HMM, an HMM with a countably infinite state space. This provides the background and context for the HDP-PCFG (Section 2).

A.1.1 Bayesian finite mixture models

We begin our tour with the familiar Bayesian finite mixture model in order to establish notation which will carry over to more complex models. The model structure is summarized in two ways in Figure 10: symbolically in the diagram on the left side of the figure and graphically in the diagram on the right side of the figure. As shown in the figure, we consider a finite mixture with K *mixture components*, where each component $z \in \{1, \ldots, K\}$ is associated with a parameter vector ϕ_z drawn from some prior distribution G_0 on a parameter space Φ. We let the vector $\beta = (\beta_1, \ldots, \beta_K)$ denote the mixing proportions; thus the probability of choosing the mixture component indexed by ϕ_z is given by β_z. This vector is assumed to be drawn from a symmetric Dirichlet distribution with hyperparameter α.

Given the parameters β and $\phi = (\phi_z : z = 1, \ldots, K)$, the data points $x = (x_1, \ldots, x_n)$ are assumed to be generated conditionally i.i.d. as follows: first choose a component z_i with probabilities given by the multinomial probability vector β and then choose x_i from the distribution indexed by z_i; i.e., from $F(\cdot; \phi_{z_i})$.

There is another way to write the finite mixture model that will be helpful in developing nonparametric Bayesian extensions. In particular, instead of expressing the choice of a mixture component as a choice of a label from a multinomial distribution, let us instead combine the mixing

proportions and the parameters associated with the mixture components into a single object—a *random measure* G—and let the choice of a mixture component be expressed as a draw from G. In particular, write

$$G = \sum_{z=1}^{K} \beta_z \delta_{\phi_z}, \tag{26}$$

where δ_{ϕ_z} is a delta function at location ϕ_z. Clearly G is random, because both the coefficients $\{\beta_z\}$ and the locations $\{\phi_z\}$ are random. It is also a measure, assigning a nonnegative real number to any Borel subset B of Φ: $G(B) = \sum_{z:\phi_z \in B} \beta_z$. In particular, as desired, G assigns probability β_z to the atom ϕ_z. To summarize, we can equivalently express the finite mixture model in Figure 10 as the draw of a random measure G (from a stochastic process that has hyperparameters G_0 and α), followed by n conditionally-independent draws of parameter vectors from G, one for each data point. Data points are then drawn from the mixture component indexed by the corresponding parameter vector.

A.1.2 Dirichlet process mixture models

The Dirichlet process (DP) mixture model extends the Bayesian finite mixture model to a mixture model having a countably infinite number of mixture components. Since we have an infinite number of mixture components, it no longer makes sense to consider a symmetric prior over the component probabilities as we did in the finite case; the prior over component probabilities must decay in some way. This is achieved via a so-called *stick-breaking distribution* (Sethuraman, 1994).

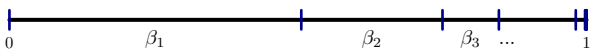


Figure 11: A sample $\beta \sim \text{GEM}(\alpha)$ from the stick-breaking distribution with $\alpha = 1$.

The stick-breaking distribution that underlies the DP mixture is defined as follows. First define a countably infinite collection of *stick-breaking proportions* $V_1, V_2, \ldots$, where the V_z are drawn independently from a one-parameter beta distribution: $V_z \sim \text{Beta}(1, \alpha)$, where $\alpha > 0$. Then define an infinite random sequence β as follows:

$$\beta_z = V_z \prod_{z'<z} (1 - V_{z'}). \tag{27}$$

As shown in Figure 11, the values of β_z defined this procedure can be interpreted as portions of a unit-length stick. In particular, the product $\prod_{z'<z}(1 - V_{z'})$ is the currently remaining portion of the stick and multiplication by V_z breaks off a proportion of the remaining stick length. It is not difficult to show that the values β_z sum to one (with probability one) and thus β can be viewed as an infinite-dimensional random probability vector.

We write $\beta \sim \text{GEM}(\alpha)$ to mean that $\beta = (\beta_1, \beta_2, \ldots)$ is distributed according to the stick-breaking distribution. The hyperparameter α determines the rate of decay of the β_z; a larger value of α implies a slower rate of decay.

We present a full specification of the DP mixture model in Figure 12. This specification emphasizes the similarity with the Bayesian finite mixture; all that has changed is that the vectors β and ϕ are infinite dimensional (note that "Mult" in Figure 12 is a multinomial distribution in an extended sense, its meaning is simply that index z is chosen with probability β_z).

Let us also consider the alternative specification of the DP mixture model using random measures. Proceeding by analogy to the finite mixture model, we define a random measure G as follows:

$$G = \sum_{z=1}^{\infty} \beta_z \delta_{\phi_z}, \tag{28}$$

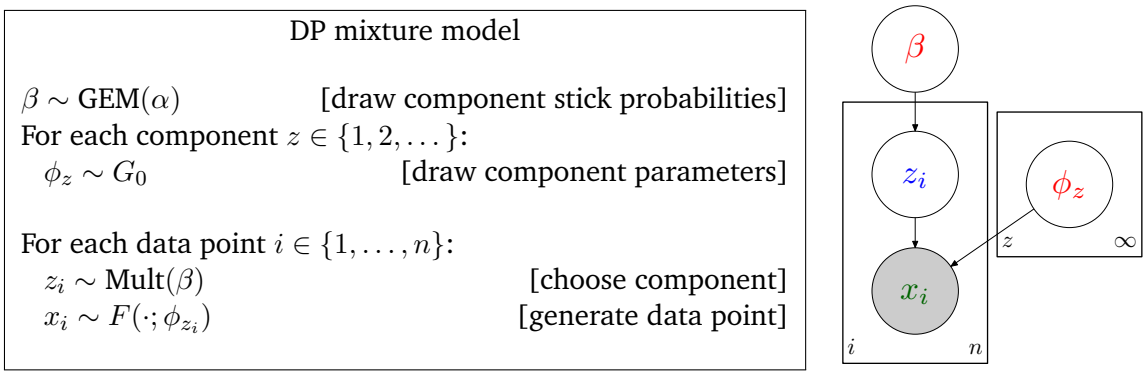


Figure 12: The definition and graphical model of the DP mixture model.

where δ_{ϕ_z} is a delta function at location ϕ_z. As before, the randomness has two sources: the random choice of the ϕ_z (which are drawn independently from G_0) and the random choice of the coefficients $\{\beta_z\}$, which are drawn from GEM(α). We say that such a random measure G is a draw from a *Dirichlet process*, denoted $G \sim \text{DP}(\alpha, G_0)$. This random measure G plays the same role in the DP mixture as the corresponding G played in the finite mixture; in particular, given G the parameter vectors are independent draws from G, one for each data point.

The term "Dirichlet process" is appropriate because the random measure G turns out to have finite-dimensional Dirichlet marginals: for an arbitrary partition $(B_1, B_2, \dots, B_r)$ of the parameter space Φ (for an arbitrary integer r), Sethuraman (1994) showed that

$$\Big(G(B_1), G(B_2), \dots, G(B_r)\Big) \sim \text{Dir}\Big(\alpha G_0(B_1), \alpha G_0(B_2), \dots, \alpha G_0(B_r)\Big).$$

A.1.3 Hierarchical Dirichlet process hidden Markov models (HDP-HMMs)

The hidden Markov model (HMM) can be viewed as a dynamic version of a finite mixture model. As in the finite mixture model, an HMM has a set of K mixture components, referred to as *states* in the HMM context. Associated to each state, $z \in \{1, 2, \dots, K\}$, is a parameter vector ϕ_z^E; this vector parametrizes a family of *emission distributions*, $F(\cdot; \phi_z^E)$, from which data points are drawn. The HMM differs from the finite mixture in that states are not selected independently, but are linked according to a Markov chain. In particular, the parametrization for the HMM includes a *transition matrix*, whose rows ϕ_z^T are the conditional probabilities of transitioning to a next state z' given that the current state is z. Bayesian versions of the HMM place priors on the parameter vectors $\{\phi_z^E, \phi_z^T\}$. Without loss of generality, assume that the initial state distribution is fixed and degenerate.

A nonparametric version of the HMM can be developed by analogy to the extension of the Bayesian finite mixture to the DP mixture. This has been done by Teh et al. (2006), following on from earlier work by Beal et al. (2002). The resulting model is referred to as the *hierarchical Dirichlet process HMM* (HDP-HMM). In this section we review the HDP-HMM, focusing on the new ingredient, which is the need for a *hierarchical* DP.

Recall that in our presentation of the DP mixture, we showed that the choice of the mixture component (i.e., the state) could be conceived in terms of a draw from a random measure G. Recall also that the HMM generalizes the mixture model by making the choice of the state conditional on the previous state. This suggests that in extending the HMM to the nonparametric setting we should consider a *set* of random measures, $\{G_z\}$, one measure for each value of the current state.

A difficulty arises, however, if we simply proceed by letting each G_z be drawn independently from a DP. In this case, the atoms forming G_z and those forming $G_{z'}$, for $z \neq z'$, will be distinct with probability one (assuming that the distribution G_0 is continuous). This means that the set of next states available from z will be entirely disjoint from the set of states available from z'.

HDP-HMM

$\beta \sim \text{GEM}(\gamma)$ [draw top-level state probabilities]
For each state $z \in \{1, 2, \dots\}$:
  $\phi_z^E \sim \text{Dir}(\gamma)$ [draw emission parameters]
  $\phi_z^T \sim \text{DP}(\alpha, \beta)$ [draw transition parameters]

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

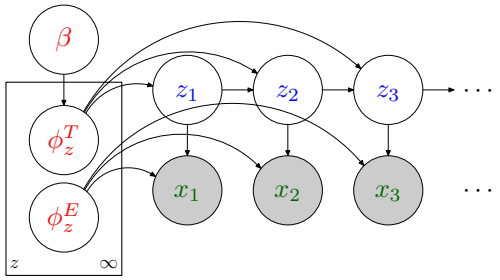


Figure 13: The definition and graphical model of the HDP-HMM.

To develop a nonparametric version of the HMM, we thus require a notion of *hierarchical Dirichlet process* (HDP), in which the random measures $\{G_z\}$ are tied. The HDP of Teh et al. (2006) does this by making a single global choice of the atoms underlying each of the random measures. Each of the individual measures G_z then weights these atoms differently. The general framework of the HDP achieves this as follows:

$$G_0 \sim \text{DP}(\gamma, H) \tag{29}$$
$$G_z \sim \text{DP}(\alpha, G_0), \quad z = 1, \dots, K, \tag{30}$$

where γ and α are concentration hyperparameters, where H is a measure and where K is the number of measures in the collection $\{G_z\}$. The key to this hierarchy is the presence of G_0 as the base measure used to draw the random measures $\{G_z\}$. Because G_0 is discrete, only the atoms in G_0 can be chosen as atoms in the $\{G_z\}$. Moreover, because the stick-breaking weights in G_0 decay, it is only a subset of the atoms—the highly-weighted atoms—that will tend to occur frequently in the measures $\{G_z\}$. Thus, as desired, we share atoms among the $\{G_z\}$.

To apply these ideas to the HMM and to the PCFG, it will prove helpful to streamline our notation. Note in particular that in the HDP specification, the same atoms are used for all of the random measures. What changes among the measures G_0 and the $\{G_z\}$ is not the atoms but the stick-breaking weights. Thus we can replace with the atoms with the positive integers and focus only on the stick-breaking weights. In particular, we re-express the HDP as follows:

$$\beta \sim \text{GEM}(\gamma) \tag{31}$$
$$\phi_z^T \sim \text{DP}(\alpha, \beta), \quad z = 1, \dots, K. \tag{32}$$

In writing the hierarchy this way, we are abusing notation. In (31), the vector β is a vector with an infinite number of components, but in (32), the symbol β refers to the measure that has atoms at the integers with weights given by the components of the vector. Similarly, the symbol ϕ_z^T refers technically to a measure on the integers, but we will also abuse notation and refer to ϕ_z^T as a vector.

It is important not to lose sight of the simple idea that is expressed by (32): the stick-breaking weights corresponding to the measures $\{G_z\}$ are reweightings of the global stick-breaking weights given by β. (An explicit formula relating ϕ_z^T and β can be found in Teh et al. (2006).)

Returning to the HDP-HMM, the basic idea is to use (31) and (32) to fill the rows of an infinite-dimensional transition matrix. In this application of the HDP, K is equal to infinity, and the vectors generated by (31) and (32) form the rows of a transition matrix for an HMM with a countably infinite state space.

We provide a full probabilistic specification of the HDP-HMM in Figure 13. Each state z is associated with transition parameters ϕ_z^T and emission parameters ϕ_z^E. Note that we have specialized

to multinomial observations in this figure, so that ϕ_z^E is a finite-dimensional vector that we endow with a Dirichlet distribution. Given the parameters $\{\phi_z^T, \phi_z^E\}$, a state sequence $(z_1, \ldots, z_n)$ and an associated observation sequence $(x_1, \ldots, x_n)$ are generated as in the classical HMM. For simplicity we assume that z_1 is always fixed to a designated START state. Given the state z_t at time t, the next state z_{t+1} is obtained by drawing an integer from $\phi_{z_t}^T$, and the observed data point at time t is obtained by a draw from $\phi_{z_t}^E$.

A.2 Bayesian inference

We would like to compute the full Bayesian posterior $p(\theta, z \mid x)$ over the HDP-PCFG grammar θ and latent parse trees z given observed sentences x. Exact computation of this posterior is intractable, so we must resort to an approximate inference algorithm.

A.2.1 Sampling versus variational inference

The two major classes of methodology available for posterior inference in the HDP-PCFG are Markov chain Monte Carlo (MCMC) sampling (Robert and Casella, 2004) or variational inference (Wainwright and Jordan, 2008). MCMC sampling is based on forming a Markov chain that has the posterior as its stationary distribution, while variational inference is based on treating the posterior distribution as the solution to an optimization problem and then relaxing that optimization problem. The two methods have complementary strengths: under appropriate conditions MCMC is guaranteed to converge to a sample from the true posterior, and its stochastic nature makes it less prone to local optima than deterministic approaches. Moreover, the sampling paradigm also provides substantial flexibility; a variety of sampling moves can be combined via Metropolis-Hastings. On the other hand, variational inference can provide a computationally-efficient approximation to the posterior. While the simplest variational methods can have substantial bias, improved approximations can be obtained (at increased computational cost) via improved relaxations. Moreover, storing and manipulating a variational approximation to a posterior can be easier than working with a collection of samples.

In our development of the HDP-PCFG, we have chosen to work within the variational inference paradigm, in part because computationally efficient inference is essential in parsing applications, and in part because of the familiarity of variational inference ideas within NLP. Indeed, the EM algorithm can be viewed as coordinate ascent on a variational approximation that is based on a degenerate posterior, and the EM algorithm (the inside-outside algorithm in the case of PCFGs (Lari and Young, 1990)) has proven to be quite effective in NLP applications to finite non-Bayesian versions of the HMM and PCFG. Our variational algorithm generalizes EM by using non-degenerate posterior distributions, allowing us to capture uncertainty in the parameters. Also, the variational algorithm is able to incorporate the DP prior, while EM cannot in a meaningful way (see Section A.3.3 for further discussion). On the other hand, the variational algorithm is procedurally very similar to EM and incurs very little additional computational overhead.

A.2.2 Representation used by the inference algorithm

Variational inference is based on an approximate representation of the posterior distribution. For DP mixture models (Figure 12), there are several possible choices of representation. In particular, one can use a stick-breaking representation (Blei and Jordan, 2005) or a *collapsed* representation based on the Chinese restaurant process, where the parameters β and ϕ have been marginalized out (Kurihara et al., 2007).

Algorithms that work in the collapsed representation have the advantage that they only need to work in the finite space of clusterings rather than the infinite-dimensional parameter space. They have also been observed to perform slightly better in some applications (Teh et al., 2007). In the

Two-component mixture model

$\phi_1 \sim \text{Beta}(1, 1)$ [draw parameter for component 1]
$\phi_2 \sim \text{Beta}(1, 1)$ [draw parameter for component 2]
For each data point $i \in \{1, \ldots, n\}$:
  $z_i \sim \text{Mult}(\frac{1}{2}, \frac{1}{2})$ [choose component]
  $x_i \sim \text{Bin}(5, \phi_{z_i})$ [generate data point]

Figure 14: A two-component mixture model.

stick-breaking representation, one must deal with the infinite-dimensional parameters by either truncating the stick (Ishwaran and James, 2001), introducing auxiliary variables that effectively provide an adaptive truncation (Walker, 2004), or adaptively allocating more memory for new components (Papaspiliopoulos and Roberts, 2008).

While collapsed samplers have been effective for the DP mixture model, there is a major drawback to using them for structured models such as the HDP-HMM and HDP-PCFG. Consider the HDP-HMM. Conditioned on the parameters, the computation of the posterior probability z can be done efficiently using the forward-backward algorithm, a dynamic program that exploits the Markov structure of the sequence. However, when parameters have been marginalizing out, the hidden states z sequence are coupled, making dynamic programming impossible.[3] As a result, collapsed samplers for the HDP-HMM generally end up sampling one state z_i at a time conditioned on the rest. This sacrifice can be especially problematic when there are strong dependencies along the Markov chain, which we would expect in natural language. For example, in an HMM model for part-of-speech tagging where the hidden states represent parts-of-speech, consider a sentence containing a two-word fragment *heads turn*. Two possible tag sequences might be NOUN VERB or VERB NOUN. However, in order for the sampler to go from one to the other, it would have to go through NOUN NOUN or VERB VERB, both of which are very low probability configurations.

An additional advantage of the non-collapsed representation is that conditioned on the parameters, inference on the parse trees decouples and can be easily parallelized, which is convenient for large datasets. Thus, we chose to use the stick-breaking representation for inference.

A.3 Mean-field variational inference for DP-based models

Since mean-field inference is only an approximation to the HDP-PCFG posterior, it is important to check that the approximation is a sensible one—that we have not entirely lost the benefits of having a nonparametric Bayesian model. For example, EM, which approximates the posterior over parameters with a single point estimate, is a poor approximation, which lacks the model selection capabilities of mean-field variational inference, as we will see in Section A.3.3.

In this section, we evaluate the mean-field approximation with some illustrative examples of simple mixture models. Section A.3.1 discusses the qualitative nature of the approximated posterior, Section A.3.2 shows how the DP prior manifests itself in the mean-field update equations, and Section A.3.3 discusses the long term effect of the DP prior over multiple mean-field iterations.

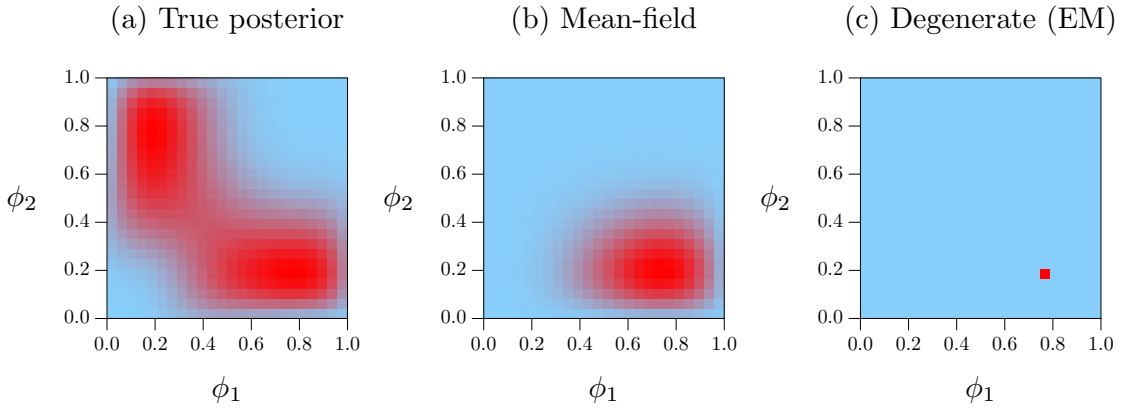


Figure 15: Shows (a) the true posterior $p(\phi_1, \phi_2 \mid x)$, (b) the optimal mean-field posterior, and (c) the optimal degenerate posterior (found by EM).

A.3.1 Mean-field approximation of the true posterior

Consider simple mixture model in Figure 14. Suppose we observe $n = 3$ data points drawn from the model: $(1, 4)$, $(1, 4)$, and $(4, 1)$, where each data point consists of 5 binomial trials. Figure 15 shows how the true posterior over parameters compares with EM and mean-field approximations. We make two remarks:

1. The true posterior is symmetrically bimodal, which reflects the non-identifiability of the mixture components: ϕ_1 and ϕ_2 can be interchanged without affecting the posterior probability. The mean-field approximation can approximate only one of those modes, but does so quite well in this simple example. In general, the mean-field posterior tends to underestimate the variance of the true posterior.

2. The mode found by mean-field has higher variance in the ϕ_1 coordinate than the ϕ_2. This is caused by the fact that there component 2 has two data points ($(1, 4)$ and $(1, 4)$) supporting the estimates whereas component 1 has only one ($(4, 1)$). EM (using MAP estimation) represents the posterior as a single point at the mode, which ignores the varying amounts of uncertainty in the parameters.

A.3.2 EM and mean-field updates in the E-step

In this section we explore the difference between EM and mean-field updates as reflected in the E-step. Recall that in the E-step, we optimize a multinomial distribution $q(z)$ over the latent discrete variables z; for the HDP-PCFG, the optimal multinomial distribution is given by (5) and is proportional to a product of multinomial weights (36). The only difference between EM and mean-field is that EM uses the maximum likelihood estimate of the multinomial parameters whereas mean-field uses the multinomial weights. Unlike the maximum likelihood solution, the multinomial weights do not have to sum to one; this provides mean-field with an additional degree of freedom that the algorithm can use to capture some of the uncertainty. This difference is manifested in two ways, via a local tradeoff and a global tradeoff between components.

Local tradeoff Suppose that $\beta \sim \text{Dir}(\alpha, \ldots, \alpha)$, and we observe counts $(c_1, \ldots, c_K) \sim \text{Mult}(\beta)$. Think of β as a distribution over the K mixture components and $c_1, \ldots, c_K$ as the expected counts

[3] However, it is still possible to sample in the collapsed representation and still exploit dynamic programming via Metropolis-Hastings (Johnson et al., 2007).

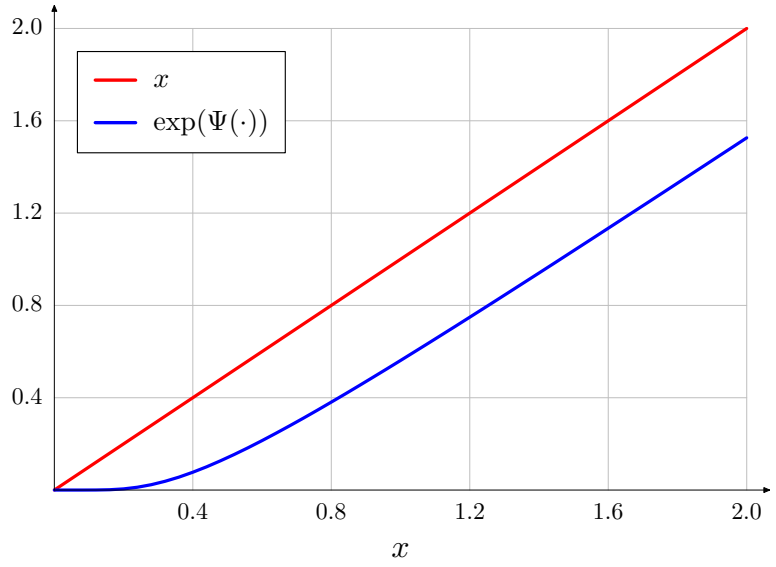


Figure 16: The $\exp(\Psi(\cdot))$ function, which is used in computing the multinomial weights for mean-field inference. It has the effect of adversely impacting small counts more than large counts.

computed in the E-step. In the M-step, we compute the posterior over β, and compute the multinomial weights needed for the next E-step.

For MAP estimation (equivalent to assuming a degenerate $q(\beta)$), the multinomial weights would be

$$W_i = \frac{c_i + \alpha - 1}{\sum_{j=1}^{K}(c_j + \alpha - 1)}.$$

When $\alpha = 1$, maximum likelihood is equivalent to MAP, which corresponds to simply normalizing the counts ($W_i \propto c_i$).

Using a mean-field approximation for $q(\beta)$ yields the following multinomial weights:

$$W_i = \exp\{E_{\beta \sim \text{Dir}(c_1+\alpha,\ldots,c_K+\alpha)} \log \beta_i\} = \frac{\exp(\Psi(c_i + \alpha))}{\exp(\Psi(\sum_{j=1}^{K}(c_j + \alpha)))}. \tag{33}$$

Let $\alpha = \frac{\alpha'}{K}$ and recall from that for large K, $\text{Dir}(\frac{\alpha'}{K}, \ldots, \frac{\alpha'}{K})$ behaves approximately like a Dirichlet process prior with concentration α' (Theorem 2 of Ishwaran and Zarepour (2002)).

When K is large, $\alpha = \frac{\alpha'}{K} \approx 0$, so, crudely, $W_i \propto \exp(\Psi(c_i))$ (although the weights need not sum to 1). The $\exp(\Psi(\cdot))$ function is shown in Figure 16. We see that $\exp(\Psi(\cdot))$ has the effect of reducing the weights; for example, when $c > \frac{1}{2}$, $\exp(\Psi(c)) \approx c - \frac{1}{2}$. However, the relative reduction $1 - \frac{\exp(\Psi(c_i))}{c_i}$ is much more for small values of c_i than for large values of c_i.

This induces a rich-gets-richer effect characteristic of Dirichlet processes, where larger counts get further boosted and small counts get further diminished. As c increases, however, the relative reduction tends to zero. This asymptotic behavior is in agreement with the general fact that the Bayesian posterior over parameters converges to a degenerate distribution at the maximum likelihood estimate.

Global tradeoff Thus far we have considered the numerator of the multinomial weight (33), which determines how the various components of the multinomial are traded off *locally*. In addition, there is a *global* tradeoff that occurs between different multinomial distributions due to the denominator. Consider the two-component mixture model from Section A.3.1. Suppose that after

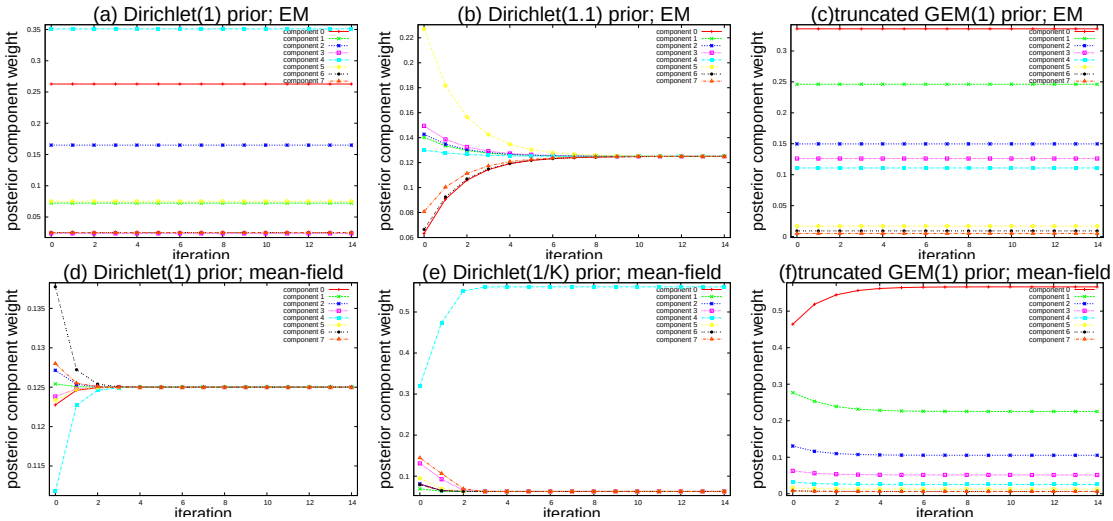


Figure 17: The evolution of the posterior mean of the component probabilities under the null data-generating distribution for three priors $p(\beta)$ using either EM or mean-field.

the E-step, the expected sufficient statistics are $(20, 20)$ for component 1 and $(0.5, 0.2)$ for component 2. In the M-step, we compute the multinomial weights W_{zj} for components $z = 1, 2$ and dimensions $j = 1, 2$. For another observation $(1, 0) \sim \text{Bin}^1(\phi_z)$, where $z \sim \text{Mult}(\frac{1}{2}, \frac{1}{2})$, the optimal posterior $q(z)$ is proportional to the multinomial weights W_{z1}.

For MAP ($q(\phi_1, \phi_2)$ is degenerate), these weights are

$$W_{11} = \frac{20}{20 + 20} = 0.5 \quad \text{and} \quad W_{21} = \frac{0.5}{0.5 + 0.2} \approx 0.714,$$

and thus component 2 has higher probability. For mean-field ($q(\phi_1, \phi_2)$ is a product of two Dirichlets), these weights are

$$W_{11} = \frac{e^{\Psi(20+1)}}{e^{\Psi(20+20+1)}} \approx 0.494 \quad \text{and} \quad W_{21} = \frac{e^{\Psi(0.5+1)}}{e^{\Psi(0.5+0.2+1)}} \approx 0.468,$$

and thus component 1 has higher probability. Note that mean-field is sensitive to the uncertainty over parameters: even though the mode of component 2 favors the observation $(1, 0)$, component 2 has sufficiently higher uncertainty that the optimal posterior $q(z)$ will favor component 1.

This global tradeoff is another way in which mean-field penalizes the use of many components. Components with more data supporting its parameters will be preferred over their meager counterparts.

A.3.3 Interaction between prior and inference

In this section we consider some of the interactions between the choice of prior (Dirichlet or Dirichlet process) and the choice of inference algorithm (EM or mean-field). Consider a K-component mixture model with component probabilities β with a "null" data-generating distribution, meaning that $F(\cdot; \cdot) \equiv 1$ (see Section A.1.1). Note that this is not the same as having zero data points, since each data point still has an unknown assignment variable z_i. Because of this, the approximate posterior $q(\beta)$ does not converge to the prior $p(\beta)$.

Figure 17 shows the impact of three priors using both EM and mean-field. Each plot shows how the posterior over components probabilities change over time. In general, the component

probabilities would be governed by both the influence of the prior shown here and the influence of a likelihood.

With a Dir$(1, \ldots, 1)$ prior and EM, the component probabilities do not change over iterations since the uniform prior over component probabilities does not prefer one component over the other (a). If we use mean-field with the same prior, the weights converge to the uniform distribution since $\alpha = 1$ roughly has the effect of adding 0.5 pseudocounts (d) (see Section A.3.2). The same asymptotic behavior can be obtained by using Dir$(1.1, \ldots, 1.1)$ and EM, which is equivalent to adding 0.1 pseudocounts (b).

Since the data-generating distribution is the same for all components, one component should be sufficient and desirable, but so far, none of these priors encourage sparsity in the components. The Dir$(\frac{1}{K}, \ldots, \frac{1}{K})$ prior (e), which approximates a DP prior (Ishwaran and Zarepour, 2002), does provide this sparsity if we use mean-field. With $\alpha = \frac{1}{K}$ near zero, components with more counts are favored. Therefore, the probability of the largest initial component increases and all others are driven close to zero. Note that EM is not well defined because the mode is not unique.

Finally, plots (c) and (f) provide two important insights:

1. The Dirichlet process is defined by using the GEM prior (Section A.1.2), but if we use EM for inference the prior does not produce the desired sparsity in the posterior (c). This is an instance where the inference algorithm is too weak to leverage the prior. The reason behind this failure is that the truncation of GEM(1) places a uniform distribution on each of the stick-breaking proportions, so there is no pressure to move away from the initial set of component probabilities.

   Note that while the truncation of GEM(1) places a uniform distribution over stick-breaking proportions, it does not induce a uniform distribution over the stick-breaking probabilities. If we were to compute the MAP stick-breaking probabilities under the induced distribution, we would obtain a different result. The reason because this discrepancy is that although the likelihood is independent of the parametrization, the prior is not. Note that full Bayesian posterior inference is independent of the parametrization.

2. If we perform approximate Bayesian inference using mean-field (f), then we can avoid the stagnation problem that we had with EM. However, the effect is not as dramatic as in (e). This shows that although the Dir$(\frac{1}{K}, \ldots, \frac{1}{K})$ prior and the truncated GEM(1) prior both converge to the DP, their effect in the context of mean-field is quite different. It is perhaps surprising that the stick-breaking prior, which places emphasis on decreasing component sizes, actually does not result in a mean-field posterior that decays as quickly as the posterior produced by the symmetric Dirichlet.

B Derivations

This section derives the update equations for the coordinate-wise ascent algorithm presented in Section 4. We begin by deriving the form of the general update (Section B.1). Then we apply this result to multinomial distributions (Section B.2) for the posterior over parse trees $q(z)$ and Dirichlet distributions (Section B.3) for the posterior over rule probabilities $q(\phi)$. Finally, we describe the optimization of the degenerate posterior over the top-level component weights $q(\beta)$ (Section B.4).

B.1 General updates

In a general setting, we have a fixed distribution $p(y)$ over the variables $y = (y_1, \ldots, y_n)$, which can be computed up to a normalization constant. (For the HDP-PCFG, $p(y) = p(z, \theta \mid x)$.) We wish to choose the best mean-field approximation $q(y) = \prod_{i=1}^{n} q(y_i)$, where the $q(y_i)$ are arbitrary

distributions. Recall that the objective of mean-field variational inference (2) is to minimize the KL-divergence between $q(y)$ and $p(y)$. Suppose we fix $q(y_{-i}) \stackrel{\text{def}}{=} \prod_{j \neq i} q(y_j)$ and wish to optimize $q(y_i)$. We rewrite the variational objective with the intent of optimizing it with respect to $q(y_i)$:

$$\begin{aligned}
\text{KL}(q(y)||p(y)) &= -E_{q(y)}[\log p(y)] - H(q(y)) \\
&= -E_{q(y)}[\log p(y_{-i})p(y_i \mid y_{-i})] - \sum_{j=1}^{n} H(q(y_j)) \\
&= \Big(- E_{q(y_i)}[E_{q(y_{-i})} \log p(y_i \mid y_{-i})] - H(q(y_i)) \Big) + \\
&\qquad \Big(- E_{q(y_{-i})} \log p(y_{-i}) - \sum_{j \neq i} H(q(y_j)) \Big) \\
&= \text{KL}(q(y_i)|| \exp\{E_{q(y_{-i})} \log p(y_i \mid y_{-i})\}) + \text{constant}, \qquad (34)
\end{aligned}$$

where the constant does not depend on $q(y_i)$. We have exploited the fact that the entropy $H(q(y))$ decomposes into a sum of individual entropies because q is fully factorized.

The KL-divergence is minimized when its two arguments are equal, so if there are no constraints on $q(y_i)$, the optimal update is given by

$$q(y_i) \propto \exp\{E_{q(y_{-i})} \log p(y_i \mid y_{-i})\}. \qquad (35)$$

In fact, we only need to compute $p(y_i \mid y_{-i})$ up to a normalization constant, because $q(y_i)$ must be normalized anyway.

From (35), we can see a strong connection between Gibbs sampling and mean-field. Gibbs sampling draws $y_i \sim p(y_i \mid y_{-i})$. Both algorithms iteratively update the approximation to the posterior distribution one variable at a time using its conditional distribution.

B.2 Multinomial updates

This general update takes us from (4) to (5) in Section 4.2.1, but we must still compute the multinomial weights defined in (6) and (7). We will show that the multinomial weights for a general Dirichlet are as follows:

$$W_i \stackrel{\text{def}}{=} \exp\{E_{\phi \sim \text{Dir}(\gamma)} \log \phi_i\} = \frac{\exp\{\Psi(\gamma_i)\}}{\exp\{\Psi(\sum_j \gamma_j)\}}, \qquad (36)$$

where $\Psi(x) = \frac{d}{dx} \log \Gamma(x)$ is the digamma function.

The core of this computation is the computation of the mean value of $\log \phi_i$ with respect to a Dirichlet distribution. Write the Dirichlet distribution in exponential family form:

$$p(\phi \mid \gamma) = \exp\Big\{ \sum_i \gamma_i \underbrace{\log \phi_i}_{\substack{\text{sufficient} \\ \text{statistics}}} \underbrace{- \Big[\sum_i \log \Gamma(\gamma_i) - \log \Gamma\Big(\sum_i \gamma_i \Big) \Big]}_{\text{log-partition function } A(\gamma)} \Big\}.$$

Since the expected sufficient statistics of an exponential family are equal to the derivatives of the cumulant function, we have

$$E \log \phi_i = \frac{\partial A(\gamma)}{\partial \gamma_i} = \Psi(\gamma_i) - \Psi\Big(\sum_j \gamma_j \Big).$$

Exponentiating both sides yields (36).

Finally, in the context of the HDP-PCFG, (36) can be applied with $\text{Dir}(\gamma)$ as $q(\phi^E)$, $q(\phi^B)$, or $q(\phi^T)$.

B.3 Dirichlet updates

Now we consider updating the Dirichlet components of the mean-field approximation, which include $q(\phi^E)$, $q(\phi^B)$, and $q(\phi^T)$. We will only derive $q(\phi^E)$ since the others are analogous. The general mean-field update (35) gives us

$$\begin{aligned}
q(\phi_z^E) &\propto \exp\Big\{E_{q(z)} \log p(\phi_z^E \mid \theta \backslash \phi_z^E, z, x)\Big\} && (37)\\
&\propto \exp\Big\{E_{q(z)} \log \prod_{x\in\Sigma} \phi_z^E(x)^{\alpha^E(x)} \prod_{x\in\Sigma} \phi_z^E(x)^{\sum_{i=1}^n \mathbb{I}[z_{[i,i]}=z, x_i=x]}\Big\} && (38)\\
&= \exp\Big\{\sum_{x\in\Sigma} E_{q(z)}\Big(\alpha^E(x) + \sum_{i=1}^n \mathbb{I}[z_{[i,i]} = z, x_i = x]\Big) \log \phi_z^E(x)\Big\} && (39)\\
&= \exp\Big\{\sum_{x\in\Sigma} (\alpha^E(x) + C^E(z,x)) \log \phi_z^E(x)\Big\} && (40)\\
&= \prod_{x\in\Sigma} \phi_z^E(x)^{\alpha^E(x)+C^E(z,x)} && (41)\\
&\propto \mathrm{Dir}(\phi_z^E; \alpha^E(\cdot) + C^E(z,\cdot)). && (42)
\end{aligned}$$

B.4 Updating top-level component weights

In this section, we discuss the computation of the objective function $L(\beta)$ and its gradient $\nabla L(\beta)$, which are required for updating the top-level component weights (Section 4.2.3).

We adapt the projected gradient algorithm (Bertsekas, 1999). At each iteration t, given the current point $\beta^{(t)}$, we compute the gradient $\nabla L(\beta^{(t)})$ and update the parameters as follows:

$$\beta^{(t+1)} \leftarrow \Pi(\beta^{(t)} + \eta_t \nabla L(\beta^{(t)})), \tag{43}$$

where η_t is the step size, which is chosen based on approximate line search, and Π projects its argument onto the K-dimensional simplex, where K is the truncation level.

Although β specifies a distribution over the positive integers, due to truncation, $\beta_z = 0$ for $z > K$ and $\beta_K = 1 - \sum_{z=1}^{K-1} \beta_z$. Thus, slightly abusing notation, we write $\beta = (\beta_1, \ldots, \beta_{K-1})$ as the optimization variables, which must reside in the set

$$\mathcal{T} = \Big\{(\beta_1, \ldots, \beta_{K-1}) : \forall 1 \le z < K, \beta_z \ge 0 \text{ and } \sum_{z=1}^{K-1} \beta_z \le 1\Big\}. \tag{44}$$

The objective function is the sum of two terms, which can be handled separately:

$$L(\beta) = \underbrace{\log \mathrm{GEM}(\beta; \alpha)}_{\text{“prior term” } L_{\text{prior}}} + \underbrace{\sum_z E_q \log \mathrm{Dir}(\phi_z^B; \alpha_B \beta\beta^\top)}_{\text{“rules term” } L_{\text{rules}}}. \tag{45}$$

Prior term Let $\pi : [0,1]^{K-1} \mapsto \mathcal{T}$ be the mapping from stick-breaking proportions $\mathbf{u} = (u_1, \ldots, u_{K-1})$ to stick-breaking weights $\beta = (\beta_1, \ldots, \beta_{K-1})$ as defined in (27). The inverse map is given by

$$\pi^{-1}(\beta) = (\beta_1/T_1, \beta_2/T_2, \ldots, \beta_{K-1}/T_{K-1})^\top,$$

where

$$T_z \stackrel{\text{def}}{=} 1 - \sum_{z'=1}^{z-1} \beta_{z'}$$

are the tail sums of the stick-breaking weights.

The density for the GEM prior is naturally defined in terms of the densities of the independent beta-distributed stick-breaking proportions $\mathbf{u}$. However, to produce a density on $\mathcal{T}$, we need to perform a change of variables. First, we compute the Jacobian of π^{-1}:

$$\nabla \pi^{-1} = \begin{pmatrix} 1/T_1 & 0 & 0 & \cdots & 0 \\ * & 1/T_2 & 0 & \cdots & 0 \\ * & * & 1/T_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ * & * & * & * & 1/T_{K-1} \end{pmatrix} \tag{46}$$

The determinant of a lower triangular matrix is the product of the diagonal entries, so we have that

$$\log \det \nabla \pi^{-1}(\beta) = -\sum_{z=1}^{K-1} \log T_z, \tag{47}$$

which yields

$$\begin{aligned} L_{\text{prior}}(\beta) &= \log \text{GEM}(\beta; \alpha) \\ &= \log \prod_{z=1}^{K-1} \text{Beta}(\beta_z/T_z; 1, \alpha) + \log \det \nabla \pi^{-1}(\beta) \\ &= \log \prod_{z=1}^{K-1} \frac{\Gamma(\alpha+1)}{\Gamma(\alpha)\Gamma(1)} (1-\beta_z/T_z)^{\alpha-1} + \log \det \nabla \pi^{-1}(\beta) \\ &= (K-1)\log\alpha + \sum_{z=1}^{K-1} (\alpha-1)\log(T_{z+1}/T_z) + \log \det \nabla \pi^{-1}(\beta) \\ &= (\alpha-1)\log T_K - \sum_{z=1}^{K-1} \log T_z + \text{constant}. \end{aligned}$$

Note that the beta prior is only applied to the first $K-1$ stick-breaking proportions, since the K-th proportion is always fixed to 1. The terms from the log densities reduce to $(\alpha-1)\log T_K$ via a telescoping sum, leaving the terms from the Jacobian as the main source of regularization.

Observing that $\frac{\partial T_z}{\partial \beta_k} = -\mathbb{I}[z > k]$, we can differentiate the prior term with respect to β_k:

$$\frac{\partial L_{\text{prior}}(\beta)}{\partial \beta_k} = -\frac{\alpha-1}{T_K} - \sum_{z=1}^{K-1} \frac{-\mathbb{I}[z>k]}{T_z} \tag{48}$$

$$= \sum_{z=k+1}^{K-1} \frac{1}{T_z} - \frac{\alpha-1}{T_K}. \tag{49}$$

To compute the effect of β_k on the objective, we only need to look at tail sums of the component weights that follow it.

For comparison, let us compute the derivative when using a Dirichlet instead of a GEM prior:

$$\tilde{L}_{\text{prior}}(\beta) = \log \text{Dir}(\beta; \alpha) = (\alpha-1)\sum_{z=1}^{K} \log \beta_k + \text{constant}. \tag{50}$$

Differentiating with respect to β_k for $k = 1, \ldots, K-1$ (β_K is a deterministic function of $\beta_1, \ldots, \beta_{K-1}$) yields:

$$\frac{\partial \tilde{L}_{\text{prior}}(\beta)}{\partial \beta_k} = (\alpha-1)\Big(\frac{1}{\beta_k} - \frac{1}{\beta_K}\Big) = \frac{\alpha-1}{\beta_k} - \frac{\alpha-1}{T_K}. \tag{51}$$

Rules term We compute the remaining term in the objective function:

$$\begin{aligned} L_{\text{rules}}(\beta) &= \sum_{z=1}^{K} E_{q(\phi)} \log \text{Dir}(\phi_z^B; \alpha_B \beta\beta^\top) \\ &= \sum_{z=1}^{K} \Big(\log\Gamma(\alpha_B) - \sum_{i=1}^{K}\sum_{j=1}^{K} \log\Gamma(\alpha_B\beta_i\beta_j) + \sum_{i=1}^{K}\sum_{j=1}^{K} (\alpha_B\beta_i\beta_j - 1) E_{q(\phi)} \log \phi_z^B(i,j) \Big). \end{aligned} \tag{52}$$

Before differentiating $L_{\text{rules}}(\beta)$, let us first differentiate a related function $L_{\text{rules}-K}(\beta, \beta_K)$, defined on K arguments instead of $K-1$:

$$\begin{aligned} \frac{\partial L_{\text{rules}-K}(\beta,\beta_K)}{\partial \beta_k} &= \sum_{z=1}^{K} \Big(-2 \sum_{i=1}^{K} \alpha_B\beta_i \Psi(\alpha_B\beta_i\beta_k) + \sum_{i=1}^{K} \alpha_B\beta_i E_{q(\phi)} \log \phi^B(k,i)\phi_z^B(i,k) \Big) \\ &= \alpha_B \sum_{z=1}^{K}\sum_{i=1}^{K} \beta_i (E_{q(\phi)} \log \phi_z^B(k,i)\phi_z^B(i,k) - 2\Psi(\alpha_B\beta_i\beta_k)). \end{aligned} \tag{53}$$

Now we can apply the chain rule using the fact that $\frac{\partial \beta_K}{\partial \beta_k} = -1$ for $k = 1, \ldots, K-1$:

$$\begin{aligned} \frac{\partial L_{\text{rules}}(\beta)}{\partial \beta_k} &= \frac{\partial L_{\text{rules}-K}(\beta,\beta_K)}{\partial \beta_k} + \frac{\partial L_{\text{rules}-K}(\beta,\beta_K)}{\partial \beta_K} \cdot \frac{\partial \beta_K}{\partial \beta_k} \\ &= \alpha_B \sum_{z=1}^{K}\sum_{i=1}^{K} \beta_i (E_{q(\phi)} \log \phi_z^B(k,i)\phi_z^B(i,k) - 2\Psi(\alpha_B\beta_i\beta_k)) - \\ &\quad \alpha_B \sum_{z=1}^{K}\sum_{i=1}^{K} \beta_i (E_{q(\phi)} \log \phi_z^B(K,i)\phi_z^B(i,K) - 2\Psi(\alpha_B\beta_i\beta_K)). \end{aligned} \tag{54}$$

References

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

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

Bertsekas, D. (1999). *Nonlinear Programming*. Belmont, MA: Athena Scientific.

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

Blunsom, P., T. Cohn, and M. Osborne (2009). Bayesian synchronous grammar induction. In *Advances in Neural Information Processing Systems (NIPS)*, Cambridge, MA. MIT Press.

Carroll, G. and E. Charniak (1992). Two experiments on learning probabilistic dependency grammars from corpora. In *Workshop Notes for Statistically-Based NLP Techniques, AAAI*, pp. 1–13.

Charniak, E. (1996). Tree-bank grammars. In *Association for the Advancement of Artificial Intelligence (AAAI)*, Cambridge, MA, pp. 1031–1036. MIT Press.

Charniak, E. (2000). A maximum-entropy-inspired parser. In *Applied Natural Language Processing and North American Association for Computational Linguistics (ANLP/NAACL)*, Seattle, Washington, pp. 132–139. Association for Computational Linguistics.

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on Information Theory 2*, 113–124.

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

DeNero, J., A. Bouchard-Côté, and D. Klein (2008). Sampling alignment structure under a Bayesian translation model. In *Empirical Methods in Natural Language Processing (EMNLP)*, Honolulu, HI, pp. 314–323.

Dyrka, W. and J. Nebel (2007). A probabilistic context-free grammar for the detection of binding sites from a protein sequence. *Systems Biology, Bioinformatics and Synthetic Biology 1*, 78–79.

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

Ferguson, T. S. (1974). Prior distributions on spaces of probability measures. *Annals of Statistics 2*, 615–629.

Finkel, J. R., T. Grenager, and C. Manning (2007). The infinite tree. In *Association for Computational Linguistics (ACL)*, Prague, Czech Republic, pp. 272–279. Association for Computational Linguistics.

Galley, M., M. Hopkins, K. Knight, and D. Marcu (2004). What's in a translation rule? In *Human Language Technology and North American Association for Computational Linguistics (HLT/NAACL)*, Boston, MA, pp. 273–280.

Gildea, D. and D. Jurafsky (2002). Automatic labeling of semantic roles. *Computational Linguistics 28*, 245–288.

Hermjakob, U. (2001). Parsing and question classification for question answering. In *Workshop on Open-domain question answering, ACL*, Toulouse, France, pp. 1–6.

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

Ishwaran, H. and M. Zarepour (2002). Exact and approximate sum-representations for the Dirichlet process. *Canadian Journal of Statististics 30*, 269–284.

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

Johnson, M., T. Griffiths, and S. Goldwater (2006). Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models. In *Advances in Neural Information Processing Systems (NIPS)*, Cambridge, MA, pp. 641–648. MIT Press.

Johnson, M., T. Griffiths, and S. Goldwater (2007). Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Human Language Technology and North American Association for Computational Linguistics (HLT/NAACL)*, Rochester, New York, pp. 139–146.

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

Klein, D. and C. D. Manning (2004). Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Association for Computational Linguistics (ACL)*, Barcelona, Spain, pp. 478–485. Association for Computational Linguistics.

Kurihara, K. 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*, Japan.

Kurihara, K., M. Welling, and Y. W. Teh (2007). Collapsed variational Dirichlet process mixture models. In *International Joint Conference on Artificial Intelligence (IJCAI)*, Hyderabad, India.

Lari, K. and S. J. Young (1990). The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language 4*, 35–56.

MacKay, D. (1997). Ensemble learning for hidden Markov models. Technical report, University of Cambridge.

Manning, C. and H. Schütze (1999). *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press.

Marcus, M. P., M. A. Marcinkiewicz, and B. Santorini (1993). Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics 19*, 313–330.

Matsuzaki, T., Y. Miyao, and J. Tsujii (2005). Probabilistic CFG with latent annotations. In *Association for Computational Linguistics (ACL)*, Ann Arbor, Michigan, pp. 75–82. Association for Computational Linguistics.

Papaspiliopoulos, O. and G. O. Roberts (2008). Retrospective MCMC for Dirichlet process hierarchical models. *Biometrika 95*, 169–186.

Pereira, F. and Y. Shabes (1992). Inside-outside reestimation from partially bracketed corpora. In *Association for Computational Linguistics (ACL)*, Newark, Delaware, pp. 128–135. Association for Computational Linguistics.

Petrov, S., 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)*, pp. 433–440. Association for Computational Linguistics.

Petrov, S. and D. Klein (2007). Learning and inference for hierarchically split PCFGs. In *Human Language Technology and North American Association for Computational Linguistics (HLT/NAACL)*, Rochester, New York, pp. 404–411.

Robert, C. P. and G. Casella (2004). *Monte Carlo Statistical Methods*. New York: Springer.

Rodriguez, A., D. B. Dunson, and A. E. Gelfand (2008). The nested Dirichlet process. *Journal of the American Statistical Association 103*, 1131–1144.

Sakakibara, Y. (2005). Grammatical inference in bioinformatics. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*, 1051–1062.

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

Smith, N. and J. Eisner (2005). Contrastive estimation: Training log-linear models on unlabeled data. In *Association for Computational Linguistics (ACL)*, Ann Arbor, Michigan, pp. 354–362. Association for Computational Linguistics.

Stolcke, A. and S. Omohundro (1994). Inducing probabilistic grammars by Bayesian model merging. In *International Colloquium on Grammatical Inference and Applications*, London, UK, pp. 106–118. Springer-Verlag.

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

Teh, Y. W., D. Newman, and M. Welling (2007). A collapsed variational Bayesian inference algorithm for Latent Dirichlet Allocation. In *Advances in Neural Information Processing Systems (NIPS)*, Cambridge, MA, pp. 1353–1360. MIT Press.

Wainwright, M. and M. I. Jordan (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning 1*, 1–307.

Walker, S. G. (2004). Sampling the Dirichlet mixture model with slices. *Communications in Statistics - Simulation and Computation 36*, 45–54.

Wu, D. (1997). Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics 23*, 377–404.

Zhu, S. C. and D. Mumford (2006). A stochastic grammar of images. *Foundations and Trends in Computer Graphics and Vision 2*, 259–362.