Variational Inference for Adaptor Grammars

Shay B. Cohen
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
scohen@cs.cmu.edu

David M. Blei
Computer Science Department
Princeton University
Princeton, NJ 08540, USA
blei@cs.princeton.edu

Noah A. Smith
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
nasmith@cs.cmu.edu

Abstract

Adaptor grammars extend probabilistic context-free grammars to define prior distributions over trees with "rich get richer" dynamics. Inference for adaptor grammars seeks to find parse trees for raw text. This paper describes a variational inference algorithm for adaptor grammars, providing an alternative to Markov chain Monte Carlo methods. To derive this method, we develop a stick-breaking representation of adaptor grammars, a representation that enables us to define adaptor grammars with recursion. We report experimental results on a word segmentation task, showing that variational inference performs comparably to MCMC. Further, we show a significant speed-up when parallelizing the algorithm. Finally, we report promising results for a new application for adaptor grammars, dependency grammar induction.

1 Introduction

Recent research in unsupervised learning for NLP focuses on Bayesian methods for probabilistic grammars (Goldwater and Griffiths, 2007; Toutanova and Johnson, 2007; Johnson et al., 2007). Such methods have been made more flexible with nonparametric Bayesian (NP Bayes) methods, such as Dirichlet process mixture models (Antoniak, 1974; Pitman, 2002). One line of research uses NP Bayes methods on whole tree structures, in the form of **adaptor grammars** (Johnson et al., 2006; Johnson, 2008b; Johnson, 2008a; Johnson and Goldwater, 2009), in order to identify recurrent subtree patterns.

Adaptor grammars provide a flexible distribution over parse trees that has more structure than a traditional context-free grammar. Adaptor grammars are used via posterior inference, the computational problem of determining the posterior distribution of parse trees given a set of observed sentences. Current posterior inference algorithms for adaptor grammars are based on MCMC sampling methods (Robert and Casella, 2005). MCMC methods are theoretically guaranteed to converge to the true posterior, but come at great expense: they are notoriously slow to converge, especially with complex hidden structures such as syntactic trees. Johnson (2008b) comments on this, and suggests the use of **variational inference** as a possible remedy.

Variational inference provides a deterministic alternative to sampling. It was introduced for Dirichlet process mixtures by Blei and Jordan (2005) and applied to infinite grammars by Liang et al. (2007). With NP Bayes models, variational methods are based on the stick-breaking representation (Sethuraman, 1994). Devising a stick-breaking representation is a central challenge to using variational inference in this setting.

The rest of this paper is organized as follows. In §2 we describe a stick-breaking representation of adaptor grammars, which enables variational inference (§3) and a well-defined incorporation of recursion into adaptor grammars. In §4 we give an empirical comparison of the algorithm to MCMC inference and describe a novel application of adaptor grammars to unsupervised dependency parsing.

2 Adaptor Grammars

We review adaptor grammars and develop a stick-breaking representation of the tree distribution.

2.1 Definition of Adaptor Grammars

Adaptor grammars capture syntactic regularities in sentences by placing a nonparametric prior over the distribution of syntactic trees that underlie them. The model exhibits "rich get richer" dynamics: once a tree is generated, it is more likely to reappear.

Adaptor grammars were developed by Johnson et al. (2006). An adaptor grammar is a tuple $\boldsymbol{A} = \langle \boldsymbol{G}, \mathcal{M}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{\alpha} \rangle$, which contains: (i) a context-free grammar $\boldsymbol{G} = \langle \mathcal{W}, \mathcal{N}, \boldsymbol{R}, S \rangle$ where $\mathcal{W}$ is the set of

terminals, $\mathcal{N}$ is the set of nonterminals, $\boldsymbol{R}$ is a set of production rules, and $S \in \mathcal{N}$ is the start symbol—we denote by $\boldsymbol{R}_A$ the subset of $\boldsymbol{R}$ with left-hand side A; (ii) a set of adapted nonterminals, $\mathcal{M} \subseteq \mathcal{N}$; and (iii) parameters $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{\alpha}$, which are described below.

An adaptor grammar assumes the following generative process of trees. First, the multinomial distributions $\boldsymbol{\theta}$ for a PCFG based on $\boldsymbol{G}$ are drawn from Dirichlet distributions. Specifically, multinomial $\boldsymbol{\theta}_A \sim \mathrm{Dir}(\alpha_A)$ where $\boldsymbol{\alpha}$ is collection of Dirichlet parameters, indexed by $A \in \mathcal{N}$.

Trees are then generated top-down starting with S. Any non-adapted nonterminal $A \in \mathcal{N} \setminus \mathcal{M}$ is expanded by drawing a rule from $\boldsymbol{R}_A$. There are two ways to expand $A \in \mathcal{M}$:

1. With probability $(n_z - b_A)/(n_A + a_A)$ we expand A to subtree z (a tree rooted at A with a yield in $\mathcal{W}^*$), where n_z is the number of times the tree z was previously generated and n_A is the total number of subtrees (tokens) previously generated root being A. We denote by $\boldsymbol{a}$ the *concentration parameters* and $\boldsymbol{b}$ the *discount parameters*, both indexed by $A \in \mathcal{M}$. We have $a_A \geq 0$ and $b_A \in [0, 1]$.
2. With probability $(a_A + k_A b_A)/(n_A + a_A)$, A is expanded as in a PCFG by a draw from $\boldsymbol{\theta}_A$ over $\boldsymbol{R}_A$, where k_A is the number of subtrees (types) previously generated with root A.

For the expansion of adapted nonterminals, this process can be explained using the Chinese restaurant process (CRP) metaphor: a "customer" (corresponding to a partially generated tree) enters a "restaurant" (corresponding to a nonterminal) and selects a "table" (corresponding to a subtree) to attach to the partially generated tree. If she is the first customer at the table, the PCFG $\langle \boldsymbol{G}, \boldsymbol{\theta} \rangle$ produces the new table's associated "dish" (a subtree).[1]

When adaptor grammars are defined using the CRP, the PCFG $\boldsymbol{G}$ has to be non-recursive with respect to the adapted nonterminals. More precisely, for $A \in \mathcal{N}$, denote by $\mathrm{Reachable}(\boldsymbol{G}, A)$ all the nonterminals that can be reached from A using a partial derivation from $\boldsymbol{G}$. Then we restrict $\boldsymbol{G}$ such that for all $A \in \mathcal{M}$, we have $A \notin \mathrm{Reachable}(\boldsymbol{G}, A)$. Without this restriction, we might end up in a situation where the generative process is ill-defined: in the CRP terminology, a customer could enter a restaurant and select a table whose dish is still in the process of being selected.[2] In the more general form of adaptor grammars with arbitrary adaptors, the problem amounts to mutually dependent definitions of distributions which rely on the others to be defined. We return to this problem in §3.1.

Inference The inference problem is to compute the posterior distribution of parse trees given observed sentences $\boldsymbol{x} = \langle x_1, \ldots, x_n \rangle$. Typically, inference with adaptor grammars is done with Gibbs sampling. Johnson et al. (2006) use an embedded Metropolis-Hastings sampler (Robert and Casella, 2005) inside a Gibbs sampler. The proposal distribution is a PCFG, resembling a tree substitution grammar (TSG; Joshi, 2003). The sampler of Johnson et al. is based on the representation of the PY process as a distribution over partitions of integers. This representation is not amenable to variational inference.

2.2 Stick-Breaking Representation

To develop a variational inference algorithm for adaptor grammars, we require an alternative representation of the model in §2.1. The CRP-based definition implicitly marginalizes out a random distribution over trees. For variational inference, we construct that distribution.

We first review the Dirichlet process and its stick-breaking representation. The Dirichlet process defines a distribution over distributions. Samples from the Dirichlet process tend to deviate from a *base distribution* depending on a *concentration parameter*. Let $G \sim \mathrm{DP}(G_0, a)$ be a distribution sampled from the Dirichlet process with base distribution G_0

[1] We note that our construction deviates from the strict definition of adaptor grammars (Johnson et al., 2006): (i) in our construction, we assume (as prior work does in practice) that the adaptors in $\boldsymbol{A} = \langle \boldsymbol{G}, \mathcal{M}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{\alpha} \rangle$ follow the Pitman-Yor (PY) process (Pitman and Yor, 1997), though in general other stochastic processes might be used; and (ii) we place a symmetric Dirichlet over the parameters of the PCFG, $\boldsymbol{\theta}$, whereas Johnson et al. used a fixed PCFG for the definition (though they experimented with a Dirichlet prior).

[2] Consider the simple grammar with rules { S → S S, S → a }. Assume that a customer enters the restaurant for S. She sits at a table, and selects a dish, a subtree, which starts with the rule S → S S. Perhaps the first child S is expanded by S → a. For the second child S, it is possible to re-enter the "S restaurant" and choose the first table, where the "dish" subtree is still being generated.

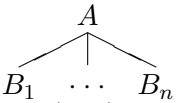
and concentration parameter a. The distribution G is discrete, which means it puts positive mass on a countable number of atoms drawn from G_0. Repeated draws from G exhibit the "clustering property," which means that they will be assigned to the same value with positive probability. Thus, they exhibit a partition structure. Marginalizing out G, the distribution of that partition structure is given by a CRP with parameter a (Pitman, 2002).

The stick-breaking process gives a constructive definition of G (Sethuraman, 1994). With the stick-breaking process (for the PY process), we first sample "stick lengths" $\boldsymbol{\pi} \sim \mathrm{GEM}(a, b)$ (in the case of Dirichlet process, we have $b = 0$). The GEM partitions the interval $[0, 1]$ into countably many segments. First, draw $v_i \sim \mathrm{Beta}(1 - b, a + ib)$ for $i \in \{1, \ldots\}$. Then, define $\pi_i \triangleq v_i \prod_{j=1}^{i-1}(1 - v_j)$. In addition, we also sample infinitely many "atoms" independently $z_i \sim G_0$. Define G as:

$$G(z) = \sum_{i=1}^{\infty} \pi_i \delta(z_i, z) \tag{1}$$

where $\delta(z_i, z)$ is 1 if $z_i = z$ and 0 otherwise. This random variable is drawn from a Pitman-Yor process. Notice the discreteness of G is laid bare in the stick-breaking construction.

With the stick-breaking representation in hand, we turn to a constructive definition of the distribution over trees given by an adaptor grammar. Let $A_1, \ldots, A_K$ be an enumeration of the nonterminals in $\mathcal{M}$ which satisfies: $i \leq j \Rightarrow A_j \notin \mathrm{Reachable}(\boldsymbol{G}, A_i)$. (That this exists follows from the assumption about the lack of recursiveness of adapted nonterminals.) Let $\mathrm{Yield}(z)$ be the yield of a tree derivation z. The process that generates observed sentences $\boldsymbol{x} = \langle x_1, \ldots, x_n \rangle$ from the adaptor grammar $\boldsymbol{A} = \langle \boldsymbol{G}, \mathcal{M}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{\alpha} \rangle$ is as follows:

1. For each $A \in \mathcal{N}$, draw $\boldsymbol{\theta}_A \sim \mathrm{Dir}(\alpha_A)$.
2. For A from A_1 to A_K, define G_A as follows:
   (a) Draw $\boldsymbol{\pi}_A \mid a_A, b_A \sim \mathrm{GEM}(a_A, b_A)$.
   (b) For $i \in \{1, \ldots\}$, grow a tree $z_{A,i}$ as follows:
      i. Draw $A \rightarrow B_1 \ldots B_n$ from $\boldsymbol{R}_A$.
      ii. $z_{A,i} =$ 

      iii. While $\mathrm{Yield}(z_{A,i})$ has nonterminals:
         A. Choose an unexpanded nonterminal B from yield of $z_{A,i}$.
         B. If $B \in \mathcal{M}$, expand B according to G_B (defined on previous iterations of step 2).
         C. If $B \in \mathcal{N} \setminus \mathcal{M}$, expand B with a rule from $\boldsymbol{R}_B$ according to $\mathrm{Mult}(\boldsymbol{\theta}_B)$.
   (c) For $i \in \{1, \ldots\}$, define $G_A(z_{A,i}) = \pi_{A,i}$
3. For $i \in \{1, \ldots, n\}$ draw z_i as follows:
   (a) If $S \in \mathcal{M}$, draw $z_i \mid G_S \sim G_S$.
   (b) If $S \notin \mathcal{M}$, draw z_i as in 2(b) (omitted for space).
4. Set $x_i = \mathrm{Yield}(z_i)$ for $i \in \{1, \ldots, n\}$.

Here, there are four collections of hidden variables: the PCFG multinomials $\boldsymbol{\theta} = \{\boldsymbol{\theta}_A \mid A \in \mathcal{N}\}$, the stick length proportions $\boldsymbol{v} = \{\boldsymbol{v}_A \mid A \in \mathcal{M}\}$ where $\boldsymbol{v}_A = \langle v_{A,1}, v_{A,2}, \ldots \rangle$, the adapted nonterminals' subtrees $\boldsymbol{z}_A = \{z_{A,i} \mid A \in \mathcal{M}; i \in \{1, \ldots\}\}$ and the derivations $\boldsymbol{z}_{1:n} = z_1, \ldots, z_n$. The symbol $\boldsymbol{z}$ refers to the collection of $\{\boldsymbol{z}_A \mid A \in \mathcal{M}\}$, and $\boldsymbol{z}_{1:n}$ refers to the derivations of the data $\boldsymbol{x}$.

Note that the distribution in 2(c) is defined with the GEM distribution, as mentioned earlier. It is a sample from the Pitman-Yor process (or the Dirichlet process), which is later used in 3(a) to sample trees for an adapted non-terminal.

3 Variational Inference

Variational inference is a deterministic alternative to MCMC, which casts posterior inference as an optimization problem (Jordan et al., 1999; Wainwright and Jordan, 2008). The optimized function is a bound on the marginal likelihood of the observations, which is expressed in terms of a so-called "variational distribution" over the hidden variables. When the bound is tightened, that distribution is close to the posterior of interest. Variational methods tend to converge faster than MCMC, and can be more easily parallelized over multiple processors in a framework such as MapReduce (Dean and Ghemawat, 2004).

The variational bound on the likelihood of the data is:

$$\begin{aligned}\log p(\boldsymbol{x} \mid \boldsymbol{a}, \boldsymbol{\alpha}) \geq H(q) &+ \sum_{A \in \mathcal{M}} \mathbb{E}_q[\log p(\boldsymbol{v}_A \mid a_A)] \\ &+ \sum_{A \in \mathcal{M}} \mathbb{E}_q[\log p(\boldsymbol{\theta}_A \mid \boldsymbol{\alpha}_A)] \\ &+ \sum_{A \in \mathcal{M}} \mathbb{E}_q[\log p(\boldsymbol{z}_A \mid \boldsymbol{v}, \boldsymbol{\theta})] + \mathbb{E}_q[\log p(\boldsymbol{z} \mid \boldsymbol{v}_A)]\end{aligned}$$

Expectations are taken with respect to the variational distribution $q(\boldsymbol{v}, \boldsymbol{\theta}, \boldsymbol{z})$ and $H(q)$ is its entropy.

Before tightening the bound, we define the functional form of the variational distribution. We use the mean-field distribution in which all of the hidden variables are independent and governed by individual variational parameters. (Note that in the true posterior, the hidden variables are highly coupled.) To account for the infinite collection of random variables, for which we cannot define a variational distribution, we use the truncated stick distribution (Blei and Jordan, 2005). Hence, we assume that, for all $A \in \mathcal{M}$, there is some value N_A such that $q(v_{A,N_A} = 1) = 1$. The assigned probability to parse trees in the stick will be 0 for $i > N_A$, so we can ignore $z_{A,i}$ for $i > N_A$. This leads to a factorized variational distribution:

$$q(\boldsymbol{v}, \boldsymbol{\theta}, \boldsymbol{z}) = \prod_{A \in \mathcal{M}} \left(q(\boldsymbol{\theta}_A) \prod_{i=1}^{N_A} q(v_{A,i}) \times q(z_{A,i}) \right) \times \prod_{i=1}^{n} q(z_i) \quad (2)$$

It is natural to define the variational distributions over $\boldsymbol{\theta}$ and $\boldsymbol{v}$ to be Dirichlet distributions with parameters $\boldsymbol{\tau}_A$ and Beta distributions with parameters $\boldsymbol{\gamma}_{A,i}$, respectively. The two distributions over trees, $q(z_{A,i})$ and $q(z_i)$, are more problematic. For example, with $q(z_i \mid \boldsymbol{\phi})$, we need to take into account different subtrees that could be generated by the model and use them with the proper probabilities in the variational distribution $q(z_i \mid \boldsymbol{\phi})$. We follow and extend the idea from Johnson et al. (2006) and use *grammatons* for these distributions. Grammatons are "mini-grammars," inspired by the grammar $\boldsymbol{G}$.

For two strings in $s, t \in \mathcal{W}^*$, we use "$t \subseteq s$" to mean that t is a substring of s. In that case, a grammaton is defined as follows:

Definition 1. *Let $\boldsymbol{A} = \langle \boldsymbol{G}, \mathcal{M}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{\alpha} \rangle$ be an adaptor grammar with $\boldsymbol{G} = \langle \mathcal{W}, \mathcal{N}, \boldsymbol{R}, S \rangle$. Let s be a finite string over the alphabet of $\boldsymbol{G}$ and $A \in \mathcal{N}$. Let $\mathcal{U}$ be the set of nonterminals $\mathcal{U} \triangleq \text{Reachable}(\boldsymbol{G}, A) \cap (\mathcal{N} \setminus \mathcal{M})$. The grammaton $\boldsymbol{G}(A, s)$ is the context-free grammar with the start symbol A and the rules*

$$R_A \cup \left(\bigcup_{B \in \mathcal{U}} R_B \right) \cup \bigcup_{A \to B_1 \ldots B_n \in R_A} \bigcup_{i \in \{i \mid B_i \in \mathcal{M}\}} \{B_i \to t \mid t \subseteq s\}.$$

Using a grammaton, we define the distributions $q(z_{A,i} \mid \boldsymbol{\phi}_A)$ and $q(z_i \mid \boldsymbol{\phi})$. This requires a preprocessing step (described in detail in §3.3) that defines, for each $A \in \mathcal{M}$, a list of strings $\boldsymbol{s}_A = \langle s_{A,1}, \ldots, s_{A,N_A} \rangle$. Then, for $q(z_{A,i} \mid \boldsymbol{\phi}_A)$ we use the grammaton $\boldsymbol{G}(A, s_{A,i})$ and for $q(z_i \mid \boldsymbol{\phi})$ we use the grammaton $\boldsymbol{G}(A, x_i)$ where x_i is the ith observed sentence. We parametrize the grammaton with weights $\boldsymbol{\phi}_A$ (or $\boldsymbol{\phi}$) for each rule in the grammaton. This makes the variational distributions over the trees for strings $\boldsymbol{s}$ (and trees for $\boldsymbol{x}$) globally normalized weighted grammars. Choosing such distributions is motivated by their ability to make the variational bound tight (similar to Cohen et al., 2008, and Cohen and Smith, 2009). In practice we do not have to use rewrite rules for all strings $t \subseteq s$ in the grammaton. It suffices to add rewrite rules only for the strings $t = s_{A,i}$ that have some grammaton attached to them, $\boldsymbol{G}(A, s_{A,i})$.

The variational distribution above yields a variational inference algorithm for approximating the posterior by estimating $\boldsymbol{\gamma}_{A,i}$, $\boldsymbol{\tau}_A$, $\boldsymbol{\phi}_A$ and $\boldsymbol{\phi}$ iteratively, given a fixed set of hyperparameters $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{\alpha}$. Let r be a PCFG rule. Let $\tilde{f}(r, s_{B,k}) = \mathbb{E}_{q(z_k \mid \boldsymbol{\phi}_{B,k})}[f(r; z_k)]$, where $f(r; z_k)$ counts the number of times that rule r is applied in the derivation z_k. Let $A \to \beta$ denote a rule from $\boldsymbol{G}$. The quantity $\tilde{f}(r, s_{B,k})$ is computed using the inside-outside (IO) algorithm. Fig. 1 gives the variational inference updates.

Variational EM We use variational EM to fit the hyperparameters. Variational EM is an EM algorithm where the E step is replaced by variational inference (Fig. 1). The M-step optimizes the hyperparameters ($\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{\alpha}$) with respect to expected sufficient statistics under the variational distribution. We use Newton-Raphson for each (Boyd and Vandenberghe, 2004); Fig. 2 gives the objectives.

3.1 Note about Recursive Grammars

With recursive grammars, the stick-breaking process representation gives probability mass to events which are ill-defined. In step 2(iii)(c) of the stick-breaking representation, we assign nonzero probability to an event in which we choose to expand the current tree using a subtree with the same index that we are currently still expanding (see footnote 2). In

short, with recursive grammars, we can get "loops" inside the trees.

We would still like to use recursion in the cases which are not ill-defined. In the case of recursive grammars, there is no problem with the stick-breaking representation and the order by which we enumerate the nonterminals. This is true because the stick-breaking process separates allocating the probabilities for each index in the stick and allocating the atoms for each index in the stick.

Our variational distributions give probability 0 to any event which is ill-defined in the sense mentioned above. Optimizing the variational bound in this case is equivalent to optimizing the same variational bound with a model p' that (i) starts with p, (ii) assigns probability 0 to ill-defined events, and (iii) renormalizes:

Proposition 2. *Let $p(\boldsymbol{x}, \boldsymbol{z})$ be a probability distribution, where $z \in \mathcal{Z}$, and let $\mathcal{S} \subset \mathcal{Z}$. Let $Q = \{q \mid q(z) = 0, \forall z \in \mathcal{S}\}$, a set of distributions. Then:*

$$\operatorname*{argmax}_{q \in Q} \mathbb{E}_q[\log p(\boldsymbol{x}, \boldsymbol{z})] \quad = \quad \operatorname*{argmax}_{q} \mathbb{E}_q[\log p'(\boldsymbol{x}, \boldsymbol{z})]$$

where $p'(\boldsymbol{x}, \boldsymbol{z})$ is a probability distribution defined as $p'(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{x}, \boldsymbol{z}) / \sum_{\boldsymbol{z} \in \mathcal{S}} p(\boldsymbol{x}, \boldsymbol{z})$ for $\boldsymbol{z} \in \mathcal{S}$ and 0 otherwise.

For this reason, our variational approximation allows the use of recursive grammars. The use of recursive grammars with MCMC methods is problematic, since it has no corresponding probabilistic interpretation, enabled by zeroing events that are ill-defined in the variational distribution. There is no underlying model such as p', and thus the inference algorithm is invalid.

3.2 Time Complexity

The algorithm in Johnson et al. (2006) works by sampling from a PCFG containing rewrite rules that rewrite to a whole tree fragment. This requires a procedure that uses the inside-outside algorithm. Despite the grammar being bigger (because of the rewrite rules to a string), the asymptotic complexity of the IO algorithm stays $O(|\mathcal{N}|^2|x_i|^3 + |\mathcal{N}|^3|x_i|^2)$ where $|x_i|$ is the length of the ith sentence.[3]

[3]This analysis is true for CNF grammars augmented with rules rewriting to a whole string, like those used in our study.

$$\begin{aligned}
\gamma^1_{A,i} &= 1 - b_A + \textstyle\sum_{B \in \mathcal{M}} \sum_{k=1}^{N_B} \tilde{f}(A \to s_{A,i}, s_{B,k}) \\
\gamma^2_{A,i} &= a_A + i b_A \\
&\quad + \textstyle\sum_{j=1}^{i-1} \sum_{B \in \mathcal{M}} \sum_{k=1}^{N_B} \tilde{f}(A \to s_{A,j}, s_{B,k}) \\
\tau_{A, A \to \beta} &= \textstyle\sum_{B \in \mathcal{M}} \sum_{k=1}^{N_B} \tilde{f}(A \to \beta, s_{B,k}) \\
\phi_{A, A \to s_{A,i}} &= \Psi(\gamma^1_{A,i}) - \Psi(\gamma^1_{A,i} + \gamma^2_{A,i}) \\
&\quad + \textstyle\sum_{j=1}^{i-1} \left(\Psi(\gamma^2_{A,i}) - \Psi(\gamma^1_{A,i} + \gamma^2_{A,i})\right) \\
\phi_{A, A \to \beta} &= \Psi(\tau_{A, A \to \beta}) - \Psi\left(\textstyle\sum_{\beta} \tau_{A, A \to \beta}\right)
\end{aligned}$$

Figure 1: Updates for variational inference with adaptor grammars. Ψ is the digamma function.

Our algorithm requires running the IO algorithm for each yield in the variational distribution, for each nonterminal, and for each sentence. However, IO runs with much smaller grammars coming from the grammatons. The cost of running the IO algorithm on the yields in the sticks for $A \in \mathcal{M}$ can be taken into account parsing a string that appears in the corpus with the full grammars. This leads to an asymptotic complexity of $O(|\mathcal{N}|^2|x_i|^3 + |\mathcal{N}|^3|x_i|^2)$ for the ith sentence in the corpus each iteration.

Asymptotically, both sampling and variational EM behave the same. However, there are different constants that hide in these asymptotic runtimes: the number of iterations that the algorithm takes to converge (for which variational EM generally has an advantage over sampling) and the number of additional rewrite rules that rewrite to a string representing a tree (for which MCMC has a relative advantage, because it does not use a fixed set of strings; instead, the size of the grammars it uses grow as sampling proceeds). In §4, we see that variational EM and sampling methods are similar in the time it takes to complete because of a trade-off between these two constants. Simple parallelization, however, which is possible only with variational inference, provides significant speed-ups.[4]

3.3 Heuristics for Variational Inference

For the variational approximation from §3, we need to decide on a set of strings, $s_{A,i}$ (for $A \in \mathcal{M}$ and $i \in \{1, \ldots, N_A\}$) to define the grammatons in the

[4]Newman et al. (2009) show how to parallelize sampling algorithms, but in general, parallelizing these algorithms is more complicated than parallelizing variational algorithms and requires further approximation.

$$\max_{\alpha_A} \quad \log\Gamma(|R_A|\alpha_A) - |R_A|\log\Gamma(\alpha_A) + (\alpha_A - 1)\left(\sum_{A\to\beta\in R_A}\Psi(\tau_{A\to\beta}) - \Psi\left(\sum_{A\to\beta\in R_A}\tau_{A\to\beta}\right)\right)$$

$$\max_{a_A} \quad \sum_{i=1}^{N_A} a_A\left(\Psi(\gamma^2_{A,i}) - \Psi(\gamma^1_{A,i}+\gamma^2_{A,i})\right) + \log\Gamma(a_A+1+ib_A) - \log\Gamma(ib_A+a_A)$$

$$\max_{b_A} \quad \sum_{i=1}^{N_A} ib_A\left(\Psi(\gamma^2_{A,i}) - \Psi(\gamma^1_{A,i}+\gamma^2_{A,i})\right) + \log\Gamma(a_A+1+ib_A) - \log\Gamma(1-b_A) - \log\Gamma(ib_A+a_A)$$

Figure 2: Variational M-step updates. Γ is the gamma function.

nonparametric stick. Any set of strings will give a valid approximation, but to make the variational approximation as accurate as possible, we require that: (i) the strings in the set must be likely to be generated using the adaptor grammar as constituents headed by the relevant nonterminal, and (ii) strings that are more likely to be generated should be associated with a lower index in the stick. The reason for the second requirement is the exponential decay of coefficients as the index increases.

We show that a simple heuristic leads to an order over the strings generated by the adaptor grammars that yields an accurate variational estimation. We begin with a weighted context-free grammar $\boldsymbol{G}_{\text{heur}}$ that has the same rules as in $\boldsymbol{G}$, only the weight for all of its rules is 1. We then compute the quantity:

$$c(A,s) = \frac{1}{n}\left(\sum_{i=1}^{n}\mathbb{E}_{\boldsymbol{G}_{\text{heur}}}[f_i(z;A,s)]\right) - \rho\log|s| \tag{3}$$

where $f_i(z;A,s)$ is a function computing the count of constituents headed by A with yield s in the tree z for the sentence x_i. This quantity can be computed by using the IO algorithm on $\boldsymbol{G}_{\text{heur}}$. The term $\rho\log|s|$ is subtracted to avoid preference for shorter constituents, similar to Mochihashi et al. (2009).

While computing $c(A,s)$ using the IO algorithm, we sort the set of all substrings of s according to their expected counts (aggregated over all strings s). Then, we use the top N_A strings in the sorted list for the grammatons of A.[5]

3.4 Decoding

The variational inference algorithm gives a distributions over parameters and hidden structures (through the grammatons). We experiment with two commonly used decoding methods: Viterbi decoding and minimum Bayes risk decoding (MBR; Goodman, 1996).

To parse a string with Viterbi (or MBR) decoding, we find the tree with highest score for the grammaton which is attached to that string. For all rules which rewrite to strings in the resulting tree, we again perform Viterbi (or MBR) decoding recursively using other grammatons.

4 Experiments

We describe experiments with variational inference for adaptor grammars for word segmentation and dependency grammar induction.

4.1 Word Segmentation

We follow the experimental setting of Johnson and Goldwater (2009), who present state-of-the-art results for inference with adaptor grammars using Gibbs sampling on a segmentation problem. We use the standard Brent corpus (Brent and Cartwright, 1996), which includes 9,790 unsegmented phonemic representations of utterances of child-directed speech from the Bernstein-Ratner (1987) corpus.

Johnson and Goldwater (2009) test three grammars for this segmentation task. The first grammar is a character unigram grammar ($\boldsymbol{G}_{\text{Unigram}}$). The second grammar is a grammar that takes into consideration collocations ($\boldsymbol{G}_{\text{Colloc}}$) which includes the rules { Sentence → Colloc, Sentence → Colloc Sentence, Colloc → Word$^+$, Word → Char$^+$ }. The third grammar incorporates more prior knowledge about the syllabic structure of English ($\boldsymbol{G}_{\text{Syllable}}$). $\boldsymbol{G}_{\text{Unigram}}$ and $\boldsymbol{G}_{\text{Syllable}}$ can be found in Johnson and Goldwater (2009). Once an utterance is parsed, Word constituents denote segments.

The value of ρ (penalty term for string length) had little effect on our results and was fixed at $\rho = -0.2$. When N_A (number of strings used in the variational distributions) is fixed, we use $N_A = 15{,}000$. We report results using Viterbi and MBR decoding. Johnson and Goldwater (2009) experimented with two

[5] The requirement to select N_A in advance is strict. We experimented with dynamic expansions of the stick, in the spirit of Kurihara et al. (2006) and Wang and Blei (2009), but we did not achieve better performance and it had an adverse effect on runtime. For completeness, we give these results in §4.

		this paper		J&G 2009	
grammar	model	Vit.	MBR	SA	MM
G^{Unigram}	Dir	0.49	**0.84**	0.57	0.54
	PY	0.49	**0.84**	0.81	0.75
	PY+inc	0.42	0.59	-	-
G^{Colloc}	Dir	0.40	**0.86**	0.75	0.72
	PY	0.40	**0.86**	0.83	**0.86**
	PY+inc	0.43	0.60	-	-
G^{Syllable}	Dir	0.77	0.83	0.84	0.84
	PY	0.77	0.83	**0.89**	0.88
	PY+inc	0.75	0.76	-	-

Table 1: F_1 performance for word segmentation on the Brent corpus. Dir. stands for Dirichlet Process adaptor ($\boldsymbol{b} = 0$), PY stands for Pitman-Yor adaptor ($\boldsymbol{b}$ optimized), and PY+inc. stands for Pitman-Yor with iteratively increasing N_A for $A \in \mathcal{M}$ (see footnote 5). J&G 2009 are the results adapted from Johnson and Goldwater (2009); SA is sample average decoding, and MM is maximum marginal decoding.

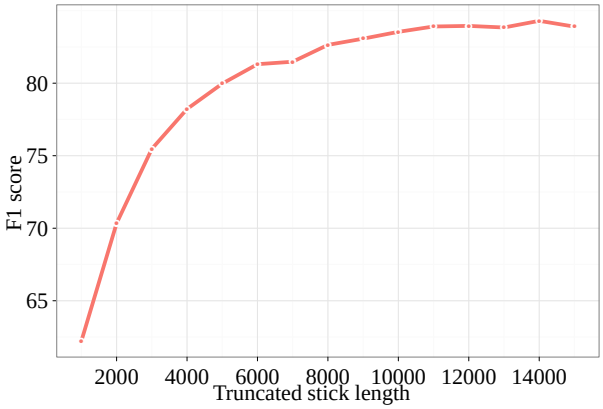


Figure 3: F_1 performance of $\boldsymbol{G}_{\text{Unigram}}$ as influenced by the length of the stick, N_{Word}.

decoding methods, sample average (SA) and maximal marginal decoding (MM), which are closely related to Viterbi and MBR, respectively. With MM, we marginalize the tree structure, rather than the word segmentation induced, similar to MBR decoding. With SA, we compute the probability of a whole tree, by averaging its count in the samples, an approximation to finding the tree with highest probability, like Viterbi.

Table 1 gives the results for our experiments. Notice that the results for the Pitman-Yor process and the Dirichlet process are similar. When inspecting the learned parameters, we noticed that the discount parameters ($\boldsymbol{b}$) learned by the variational inference algorithm for the Pitman-Yor process are very close to 0. In this case, the Pitman-Yor process is reduced to the Dirichlet process.

Similar to Johnson and Goldwater's comparisons, we see superior performance when using minimum Bayes risk over Viterbi decoding. Further notice that the variational inference algorithm obtains significantly superior performance for simpler grammars than Johnson et al., while performance using the syllable grammar is lower. The results also suggest that it is better to decide ahead on the set of strings available in the sticks, instead of working gradually and increase the size of the sticks as described in footnote 5. We believe that the reason is that the variational inference algorithm settles in a trajectory that uses fewer strings, then fails to exploit the strings that are added to the stick later. Given that selecting N_A in advance is advantageous, we may inquire if choosing N_A to be too large can lead to degraded performance, because of fragmention of the grammar. Fig. 3 suggests it is not the case, and performance stays steady after N_A reaches a certain value.

One of the advantages of variational approximation over sampling methods is the ability to run for fewer iterations. For example, with $\boldsymbol{G}_{\text{Unigram}}$ convergence typically takes 40 iterations with variational inference, while Johnson and Goldwater (2009) ran their sampler for 2,000 iterations, for which 1,000 were for burning in. The inside-outside algorithm dominates the iteration's runtime, both for sampling and variational EM. Each iteration with sampling, however, takes less time, despite the asymptotic analysis in §3.2, because of different implementations and the different number of rules that rewrite to a string. We now give a comparison of clock time for $\boldsymbol{G}_{\text{Unigram}}$ for variational inference and sampling as described in Johnson and Goldwater (2009).[6] Replicating the experiment in Johnson and Goldwater (first row in Table 1) took 2 hours and 11 minutes. With the variational approximation, we had the following: (i) the preprocessing (§3.3) step took 114 seconds; (ii) each iteration took approximately 204 seconds, with convergence after 40 iterations, leading to 8,160 seconds of pure varia-

[6] We used the code and data available at `http://www.cog.brown.edu/~mj/Software.htm`. The machine used for this comparison is a 64-bit machine with 2.6GHz CPU, 4MB of cache memory and 8GB of RAM.

tional EM processing; (iii) parsing took another 952 seconds. The total time is 2 hours and 34 minutes.

At first glance it seems that variational inference is slower than MCMC sampling. However, note that the cost of the grammar preprocessing step is amortized over all experiments with the specific grammar, and the E-step with variational inference can be parallelized, while sampling requires an update of a global set of parameters after each tree update. We ran our algorithm on a cluster of 20 1.86GHz CPUs and achieved a significant speed-up: preprocessing took 34 seconds, each variational EM iteration took 43 seconds and parsing took 208 seconds. The total time was 47 minutes, which is 2.8 times faster than sampling.

4.2 Dependency Grammar Induction

We conclude our experiments with preliminary results for unsupervised syntax learning. This is a new application of adaptor grammars, which have so far been used in segmentation (Johnson and Goldwater, 2009) and named entity recognition (Elsner et al., 2009).

The grammar we use is the dependency model with valence (DMV Klein and Manning, 2004) represented as a probabilistic context-free grammar, $\boldsymbol{G}_{\mathrm{DMV}}$ (Smith, 2006). We note that $\boldsymbol{G}_{\mathrm{DMV}}$ is recursive; this is not a problem (§3.1).

We used part-of-speech sequences from the *Wall Street Journal* Penn Treebank (Marcus et al., 1993), stripped of words and punctuation. We follow standard parsing conventions and train on sections 2–21 and test on section 23 (while using sentences of length 10 or less). Because of the unsupervised nature of the problem, we report results on the training set, in addition to the test set.

The nonterminals that we adapted correspond to nonterminals that define noun constituents. We then use the preprocessing step defined in §3.3 with a uniform grammar and take the top 3,000 strings for each nonterminal of a noun constituent.

The results are in Table 4.2. We report attachment accuracy, the fraction of parent-child relationships that the algorithm classified correctly. Notice that the results are not very different for Viterbi and MBR decoding, unlike the case with word segmentation. It seems like the DMV grammar, applied to this task, is more robust to changes in decoding mechanism. Adaptor grammars improve performance over classic EM and variational EM with a Dirichlet prior significantly.

	model	Vit.	MBR
train	non-Bayesian	48.2	48.3
	Dirichlet prior	48.3	48.6
	Adaptor grammar	**54.0**	†53.7
test	non-Bayesian	45.8	46.1
	Dirichlet prior	45.9	46.1
	Adaptor grammar	48.3	**50.2**

Table 2: Attachment accuracy for different models for dependency grammar induction. Bold marks best overall accuracy per evaluation set, and † marks figures that are not significantly worse (binomial sign test, $p < 0.05$).

We note that adaptor grammars are not limited to a selection of a Dirichlet distribution as a prior for the grammar rules. Our variational inference algorithm, for example, can be extended to use the logistic normal prior instead of the Dirichlet, shown successful by Cohen and Smith (2009).[7]

5 Conclusion

We described a variational inference algorithm for adaptor grammars based on a stick-breaking process representation, which solves a problem with adaptor grammars and recursive PCFGs. We tested it for a segmentation task, and showed results which are either comparable or an imporvement of state of the art. We showed that significant speed-ups can be obtained using parallelization of the algorithm. We also tested the algorithm on a novel task for adaptor grammars, dependency grammar induction. We showed that an improvement can be obtained using adaptor grammars over non-Bayesian and parametric baselines.

Acknowledgments

The authors would like to thank the anonymous reviewers, Jordan Boyd-Graber, Reza Haffari, Mark Johnson, and Chong Wang for their useful feedback and comments. This work was supported by the following grants: ONR 175-6343 and NSF CAREER 0745520 to Blei; NSF IIS-0836431 and IIS-0915187 to Smith.

[7] The performance of Cohen and Smith (2009), like the performance of Headden et al. (2009), is greater than what we report, but those developments are orthogonal to the contributions of this paper.

References

C. Antoniak. 1974. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics*, 2(6):1152–1174.

N. Bernstein-Ratner. 1987. The phonology of parent child speech. *Children's Language*, 6.

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

S. Boyd and L. Vandenberghe. 2004. *Convex Optimization*. Cambridge Press.

M. Brent and T. Cartwright. 1996. Distributional regularity and phonotactic constraints are useful for segmentation. *Cognition*, 6:93–125.

S. B. Cohen and N. A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proc. of NAACL-HLT*.

S. B. Cohen, K. Gimpel, and N. A. Smith. 2008. Logistic normal priors for unsupervised probabilistic grammar induction. In *NIPS*.

J. Dean and S. Ghemawat. 2004. MapReduce: Simplified data processing on large clusters. In *Proc. of OSDI*.

M. Elsner, E. Charniak, and M. Johnson. 2009. Structured generative models for unsupervised named-entity clustering. In *Proc. of NAACL-HLT*.

S. Goldwater and T. L. Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proc. of ACL*.

J. Goodman. 1996. Parsing algorithms and metrics. In *Proc. of ACL*.

W. P. Headden, M. Johnson, and D. McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proc. of NAACL-HLT*.

M. Johnson and S. Goldwater. 2009. Improving nonparameteric Bayesian inference experiments on unsupervised word segmentation with adaptor grammars. In *Proc. of NAACL-HLT*.

M. Johnson, T. L. Griffiths, and S. Goldwater. 2006. Adaptor grammars: A framework for specifying compositional nonparameteric Bayesian models. In *NIPS*.

M. Johnson, T. L. Griffiths, and S. Goldwater. 2007. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Proc. of NAACL*.

M. Johnson. 2008a. Unsupervised word segmentation for Sesotho using adaptor grammars. In *Proceedings of the Tenth Meeting of ACL Special Interest Group on Computational Morphology and Phonology*.

M. Johnson. 2008b. Using adaptor grammars to identify synergies in the unsupervised acquisition of linguistic structure. In *Proc. of ACL*.

M. I. Jordan, Z. Ghahramani, T. S. Jaakola, and L. K. Saul. 1999. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233.

A. Joshi. 2003. Tree adjoining grammars. In R. Mitkov, editor, *The Oxford Handbook of Computational Linguistics*, pages 483–501. Oxford University Press.

D. Klein and C. D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proc. of ACL*.

K. Kurihara, M. Welling, and N. A. Vlassis. 2006. Accelerated variational Dirichlet process mixtures. In *NIPS*.

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

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

D. Mochihashi, T. Yamada, and N. Ueda. 2009. Bayesian unsupervised word segmentation with nested Pitman-Yor language modeling. In *Proc. of ACL*.

D. Newman, A. Asuncion, P. Smyth, and M. Welling. 2009. Distributed algorithms for topic models. *Journal of Machine Learning Research*, 10:1801–1828.

J. Pitman and M. Yor. 1997. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *Annals of Probability*, 25(2):855–900.

J. Pitman. 2002. *Combinatorial Stochastic Processes*. Lecture Notes for St. Flour Summer School. Springer-Verlag, New York, NY.

C. P. Robert and G. Casella. 2005. *Monte Carlo Statistical Methods*. Springer.

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

N. A. Smith. 2006. *Novel Estimation Methods for Unsupervised Discovery of Latent Structure in Natural Language Text*. Ph.D. thesis, Johns Hopkins University.

K. Toutanova and M. Johnson. 2007. A Bayesian LDA-based model for semi-supervised part-of-speech tagging. In *Proc. of NIPS*.

M. J. Wainwright and M. I. Jordan. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1:1–305.

C. Wang and D. M. Blei. 2009. Variational inference for the nested Chinese restaurant process. In *NIPS*.