Online Adaptor Grammars with Hybrid Inference

Ke Zhai  
Computer Science and UMIACS  
University of Maryland  
College Park, MD USA  
zhaike@cs.umd.edu

Jordan Boyd-Graber  
Computer Science  
University of Colorado  
Boulder, CO USA  
jordan.boyd.graber@colorado.edu

Shay B. Cohen  
School of Informatics  
University of Edinburgh  
Edinburgh, Scotland, UK  
scohen@inf.ed.ac.uk

Abstract

Adaptor grammars are a flexible, powerful formalism for defining nonparametric, unsupervised models of grammar productions. This flexibility comes at the cost of expensive inference. We address the difficulty of inference through an online algorithm which uses a hybrid of Markov chain Monte Carlo and variational inference. We show that this inference strategy improves scalability without sacrificing performance on unsupervised word segmentation and topic modeling tasks.

1 Introduction

Nonparametric Bayesian models are effective tools to discover latent structure in data (Müller and Quintana, 2004). These models have had great success in text analysis, especially syntax (Shindo et al., 2012). Nonparametric distributions provide support over a countably infinite long-tailed distributions common in natural language (Goldwater et al., 2011).

We focus on adaptor grammars (Johnson et al., 2006), syntactic nonparametric models based on probabilistic context-free grammars. Adaptor grammars weaken the strong statistical independence assumptions PCFGs make (Section 2).

The weaker statistical independence assumptions that adaptor grammars make come at the cost of expensive inference. Adaptor grammars are not alone in this trade-off. For example, nonparametric extensions of topic models (Teh et al., 2006) have substantially more expensive inference than their parametric counterparts (Yao et al., 2009).

A common approach to address this computational bottleneck is through variational inference (Wainwright and Jordan, 2008). One of the advantages of variational inference is that it can be easily parallelized (Nallapati et al., 2007) or transformed into an online algorithm (Hoffman et al., 2010), which often converges in fewer iterations than batch variational inference.

Past variational inference techniques for adaptor grammars assume a preprocessing step that looks at all available data to establish the support of these nonparametric distributions (Cohen et al., 2010). Thus, these past approaches are not directly amenable to online inference.

Markov chain Monte Carlo (MCMC) inference, an alternative to variational inference, does not have this disadvantage. MCMC is easier to implement, and it discovers the support of nonparametric models during inference rather than assuming it a priori.

We apply stochastic hybrid inference (Mimno et al., 2012) to adaptor grammars to get the best of both worlds. We interleave MCMC inference inside variational inference. This preserves the scalability of variational inference while adding the sparse statistics and improved exploration MCMC provides.

Our inference algorithm for adaptor grammars starts with a variational algorithm similar to Cohen et al. (2010) and adds hybrid sampling within variational inference (Section 3). This obviates the need for expensive preprocessing and is a necessary step to create an online algorithm for adaptor grammars.

Our online extension (Section 4) processes examples in small batches taken from a stream of data. As data arrive, the algorithm dynamically extends the underlying approximate posterior distributions as more data are observed. This makes the algorithm flexible, scalable, and amenable to datasets that cannot be examined exhaustively because of their size—e.g., terabytes of social media data appear every second—or their nature—e.g., speech acquisition, where a language learner is limited to the bandwidth of the human perceptual system and cannot acquire data in a monolithic batch (Börschinger and Johnson, 2012).

We show our approach’s scalability and effective-
ness by applying our inference framework in Section 5 on two tasks: unsupervised word segmentation and infinite-vocabulary topic modeling.

2 Background

In this section, we review probabilistic context-free grammars and adaptor grammars.

2.1 Probabilistic Context-free Grammars

Probabilistic context-free grammars (PCFG) define probability distributions over derivations of a context-free grammar. We define a PCFG $G$ to be a tuple $\langle W, N, R, S, \theta \rangle$: a set of terminals $W$, a set of nonterminals $N$, productions $R$, start symbol $S \in N$ and a vector of rule probabilities $\theta$. The rules that rewrite nonterminal $c$ is $R(c)$. For a more complete description of PCFGs, see Manning and Schütze (1999).

PCFGs typically use nonterminals with a syntactic interpretation. A sequence of terminals (the yield) is generated by recursively rewriting nonterminals as sequences of child symbols (either a nonterminal or a symbol). This builds a hierarchical phrase-tree structure for every yield.

For example, a nonterminal VP represents a verb phrase, which probabilistically writes into a sequence of nonterminals $V, N$ (corresponding to verb and noun) using the production rule $VP \rightarrow V N$. Both nonterminals can be further rewritten. Each nonterminal has a multinomial distribution over expansions; for example, a multinomial for nonterminal $N$ would rewrite as “cake”, with probability $\theta_{N \rightarrow \text{cake}} = 0.03$. Rewriting terminates when the derivation has reached a terminal symbol such as “cake” (which does not rewrite).

While PCFGs are used both in the supervised setting and in the unsupervised setting, in this paper we assume an unsupervised setting, in which only terminals are observed. Our goal is to predict the underlying phrase-structure tree.

2.2 Adaptor Grammars

PCFGs assume that the rewriting operations are independent given the nonterminal. This context-freeness assumption often is too strong for modeling natural language.

Adaptor grammars break this independence assumption by transforming a PCFG’s distribution over trees $G_c$ rooted at nonterminal $c$ into a richer distribution $H_c$ over the trees headed by a nonterminal $c$, which is often referred to as the grammaton.

A Pitman-Yor Adaptor grammar (PYAG) forms the adapted tree distributions $H_c$, using a Pitman-Yor process (Pitman and Yor, 1997, PY), a generalization of the Dirichlet process (Ferguson, 1973, DP). A draw $H_c \equiv (\pi_c, z_c)$ is formed by the stick-breaking process (Sudderth and Jordan, 2008, PYGEM) parametrized by scale parameter $\alpha$, discount factor $b$, and base distribution $G_c$:

\[
\begin{align*}
\pi_k & \sim \text{Beta}(1 - b, \alpha + kb), \quad z_k \sim G_c, \\
\pi_k & \equiv \pi'_k \prod_{j=1}^{k-1} (1 - \pi'_j), \quad H \equiv \sum_k \pi_k \delta_{z_k}. \tag{1}
\end{align*}
\]

Intuitively, the distribution $H_c$ is a discrete reconstruction of the atoms sampled from $G_c$—hence, reweights $G_c$. Grammaton $H_c$ assigns non-zero stick-breaking weights $\pi$ to a countably infinite number of parse trees $z$. We describe learning these grammatons in Section 3.

More formally, a PYAG is a quintuple $\mathcal{A} = (G, M, a, b, \alpha)$ with: a PCFG $G$; a set of adapted nonterminals $M \subseteq N$; Pitman-Yor process parameters $a_c, b_c$ at each adaptor $c \in M$ and Dirichlet parameters $\alpha_c$ for each nonterminal $c \in N$. We also assume an order on the adapted nonterminals, $c_1, \ldots, c_{|M|}$ such that $c_j$ is not reachable from $c_i$ in a derivation if $j > i$.

Algorithm 1 describes the generative process of an adaptor grammar on a set of $D$ observed sentences $x_1, \ldots, x_D$.

---

1Adaptor grammars, in their general form, do not have to use the Pitman-Yor process but have only been used with the Pitman-Yor process.

2This is possible because we assume that recursive nonterminals are not adapted.
Given a PYAG $A$, the joint probability for a set of sentences $X$ and its collection of trees $T$ is

$$p(X, T, \pi, \theta, z|A) = \prod_{c \in M} p(\pi_c|a_c, b_c)p(z_c|G_c) \prod_{c \in N} p(\theta_c|\alpha_c) \prod_{x_d \in X} p(x_d|\theta, \pi, z),$$

where $x_d$ and $t_d$ represent the $d^{th}$ observed string and its corresponding parse. The multinomial PCFG parameter $\theta_c$ is drawn from a Dirichlet distribution at nonterminal $c \in N$. At each adapted nonterminal $c \in M$, the stick-breaking weights $\pi_c$ are drawn from a PYGEM (Equation 1). Each weight has an associated atom $z_c,i$ from base distribution $G_c$, a subtree rooted at $c$. The probability $p(x_d, t_d|\theta, \pi, z)$ is the PCFG likelihood of yield $x_d$ with parse tree $t_d$.

Adaptor grammars require a base PCFG such that it does not have recursive adapted nonterminals, i.e., there cannot be a path in a derivation from a given adapted nonterminal to a second appearance of that adapted nonterminal.

## 3 Hybrid Variational-MCMC Inference

Discovering the latent variables of the model—trees, adapted probabilities, and PCFG rules—is a problem of posterior inference given observed data. Previous approaches use MCMC (Johnson et al., 2006) or variational inference (Cohen et al., 2010).

MCMC discovers the support of nonparametric models during the inference, but does not scale to larger datasets (due to tight coupling of variables). Variational inference, however, is inherently parallel and easily amendable to online inference, but requires preprocessing to discover the adapted productions. We combine the best of both worlds and propose a hybrid variational-MCMC inference algorithm for adaptor grammars.

Variational inference posits a variational distribution over the latent variables in the model; this in turn induces an “evidence lower bound” (ELBO, $\mathcal{L}$) as a function of a variational distribution $q$, a lower bound on the marginal log-likelihood. Variational inference optimizes this objective function with respect to the parameters that define $q$.

In this section, we derive coordinate-ascent updates for these variational parameters. A key mathematical component is taking expectations with respect to the variational distribution $q$. We strategically use MCMC sampling to compute the expectation of $q$ over parse trees $z$. Instead of explicitly computing the variational distribution for all parameters, one can sample from it. This produces a sparse approximation of the variational distribution, which improves both scalability and performance. Sparse distributions are easier to store and transmit in implementations, which improves scalability. Mimno et al. (2012) also show that sparse representations improve performance. Moreover, because it can flexibly adjust its support, it is a necessary prerequisite to online inference (Section 4).

### 3.1 Variational Lower Bound

We posit a mean-field variational distribution:

$$q(\pi, \theta, T|\gamma, \nu, \phi) = \prod_{c \in M} \prod_{i=1}^{\infty} q(\pi_{c,i}^1|\nu_{c,i}^1, \nu_{c,i}^2) \prod_{c \in N} q(\theta_c|\gamma_c) \prod_{x_d \in X} q(t_d|\phi_d),$$

where $\pi_{c,i}^1$ is drawn from a variational Beta distribution parameterized by $\nu_{c,i}^1, \nu_{c,i}^2$; and $\theta_c$ is from a variational Dirichlet prior $\gamma_c \in \mathbb{R}_+^{[R(c)]}$. Index $i$ ranges over a possibly infinite number of adapted rules. The parse for the $d^{th}$ observation, $t_d$ is modeled by a multinomial $\phi_d$, where $\phi_{d,i}$ is the probability generating the $i^{th}$ phrase-structure tree $t_{d,i}$.

The variational distribution over latent variables induces the following ELBO on the likelihood:

$$\mathcal{L}(z, \pi, \theta, T, D; a, b, \alpha) = H[q(\theta, \pi, T)]$$

\begin{align}
&+ \sum_{c \in N} \mathbb{E}_q[\log p(\theta_c|\alpha_c)] \\
&+ \sum_{c \in M} \sum_{i=1}^{\infty} \mathbb{E}_q[\log p(\pi_{c,i}^1|a_c, b_c)] \\
&+ \sum_{c \in M} \sum_{i=1}^{\infty} \mathbb{E}_q[\log p(z_{c,i}|\pi, \theta)] \\
&+ \sum_{x_d \in X} \mathbb{E}_q[\log p(x_d, t_d|\theta, \pi, z)],
\end{align}

where $H[\bullet]$ is the entropy function.

To make this lower bound tractable, we truncate the distribution over $\pi$ to a finite set (Blei and Jordan, 2005) for each adapted nonterminal $c \in M$, i.e., $\pi_{c,K_c} \equiv \mathbb{I}$ for some index $K_c$. Because the atom weights $\pi_{t,k}$ are deterministically defined by Equation 1, this implies that $\pi_{c,i}$ is zero beyond index $K_c$. Each weight $\pi_{c,i}$ is associated with an atom $z_{c,i}$, a subtree rooted at $c$. We call the ordered set of $z_{c,i}$ the truncated nonterminal grammarn (TNG). Each adapted nonterminal $c \in M$ has its own TNG$_c$. The $i^{th}$ subtree in TNG$_c$ is denoted TNG$_{c,i}$.
In the rest of this section, we describe approximate inference to maximize $L$. The most important update is $\phi_{d,i}$, which we update using stochastic MCMC inference (Section 3.2). Past variational approaches for adaptor grammars (Cohen et al., 2010) rely on a preprocessing step and heuristics to define a static TNG. In contrast, our model dynamically discovers trees. The TNG grows as the model sees more data, allowing online updates (Section 4).

The remaining variational parameters are optimized using expected counts of adaptor grammar rules. These expected counts are described in Section 3.3, and the variational updates for the variational parameters excluding $\phi_{d,i}$ are described in Section 3.4.

### 3.2 Stochastic MCMC Inference

Each observation $x_d$ has an associated variational multinomial distribution $\phi_d$ over trees $t_d$ that can yield observation $x_d$ with probability $\phi_{d,i}$. Holding all other variational parameters fixed, the coordinate-ascent update (Mimno et al., 2012; Bishop, 2006) for $\phi_{d,i}$ is

$$\phi_{d,i} \propto \exp\{E_q[\log p(t_d, x_d, \pi, \theta, z)]\}, \quad (4)$$

where $\phi_{d,i}$ is the probability generating the $i$th phrase-structure tree $t_{d,i}$ and $E_q[\phi_d]$ is the expectation with respect to the variational distribution $q$, excluding the value of $\phi_d$.

Instead of computing this expectation explicitly, we turn to stochastic variational inference (Börschinger and Johnson, 2012) for adaptor grammars (Cohen et al., 2010) to sample from this distribution. This produces a set of sampled trees $\sigma_d = \{\sigma_{d,1}, \ldots, \sigma_{d,k}\}$. From this set of trees we can approximate our variational distribution over trees $\phi$ using the empirical distribution $\sigma_d$, i.e.,

$$\phi_{d,i} \propto I[\sigma_{d,j} = t_{d,i}, \forall \sigma_{d,j} \in \sigma_d]. \quad (5)$$

This leads to a sparse approximation of variational distribution $\phi$.\(^3\)

Previous inference strategies (Johnson et al., 2006; Börschinger and Johnson, 2012) for adaptor grammars have used sampling. The adaptor grammar inference methods use an approximate PCFG to emulate the marginalized Pitman-Yor distributions at each nonterminal. Given this approximate PCFG, we can then sample a derivation $z$ for string $x$ from the possible trees (Johnson et al., 2007).

Sampling requires a derived PCFG $\mathcal{G}'$ that approximates the distribution over tree derivations conditioned on a yield. It includes the original PCFG rules $\mathcal{R} = \{c \rightarrow \beta\}$ that define the base distribution and the new adapted productions $\mathcal{R}' = \{c \Rightarrow z, z \in \text{TNG}_c\}$. Under $\mathcal{G}'$, the probability $\theta'$ of adapted production $c \Rightarrow z$ is

$$\log \theta'_{c \Rightarrow z} = \begin{cases} E_q[\log \pi_c], & \text{if } \text{TNG}_c(i) = z \\ E_q[\log \pi_{c,K_c}] + E_q[\log \theta_{c \Rightarrow z}], & \text{otherwise} \end{cases}, \quad (6)$$

where $K_c$ is the truncation level of $\text{TNG}_c$ and $\pi_{c,K_c}$ represents the left-over stick weights in the stick-breaking process for adaptor $c \in M$. $\theta_{c \Rightarrow z}$ represents the probability of generating tree $c \Rightarrow z$ under the base distribution. See also Cohen (2011).

The expectation of the Pitman-Yor multinomial $\pi_{c,i}$ under the truncated variational stick-breaking distribution is

$$E_q[\log \pi_{a,i}] = \Psi(\nu_{a,i}^1) - \Psi(\nu_{a,i}^1 + \nu_{a,i}^2) + \sum_{j=1}^{i-1}(\Psi(\nu_{a,j}^2) - \Psi(\nu_{a,j}^1 + \nu_{a,j}^2), \quad (7)$$

and the expectation of generating the phrase-structure tree $a \Rightarrow z$ based on PCFG productions under the variational Dirichlet distribution is

$$E_q[\log \theta_{a \Rightarrow z}] = \sum_{c \rightarrow \beta \rightarrow a \Rightarrow z} (\Psi(\gamma_{c \rightarrow \beta}) - \Psi(\sum_{c \rightarrow \beta' \in R_c, \gamma_{c \rightarrow \beta'}})) \quad (8)$$

where $\Psi(\bullet)$ is the digamma function, and $c \rightarrow \beta \in a \Rightarrow z$ represents all PCFG productions in the phrase-structure tree $a \Rightarrow z$.

This PCFG can compose arbitrary subtrees and thus discover new trees that better describe the data, even if those trees are not part of the TNG. This is equivalent to creating a “new table” in MCMC inference and provides truncation-free variational updates (Wang and Blei, 2012) by sampling an unseen subtree with adapted nonterminal $c \in M$ at the root. This frees our model from preprocessing to initialize truncated grammaton in Cohen et al. (2010). This stochastic approach has the advantage of creating sparse distributions (Wang and Blei, 2012): few unique trees will be represented.

\(^3\)In our experiments, we use ten samples.
Figure 1: Given an adaptor grammar, we sample derivations via the Chinese restaurant metaphor (Johnson et al., 2006). Existing cached rules (elements in the TNG) can be thought of as occupied tables; this happens in the case of the yield “ba”, which increases counts for unadapted rules \( g \) and for entries in \( \text{TNG}_A \), \( f \). For the yield “ca”, there is no appropriate entry in the TNG, so it must use the base distribution, which corresponds to sitting at a new table. This generates counts for \( g \), as it uses the unadapted rule and for \( h \), which represents entries that could be included in the TNG in the future. The final yield, “ab”, shows that even when compatible entries are in the TNG, it might still create a new table, changing the underlying base distribution.

### Parallelization

As noted in Cohen et al. (2010), the inside-outside algorithm dominates the runtime of every iteration, both for sampling and variational inference. However, unlike MCMC, variational inference is highly parallelizable and requires fewer synchronizations per iteration (Zhai et al., 2012). In our approach, both inside algorithms and sampling process can be distributed, and those counts can be aggregated afterwards. In our implementation, we use multiple threads to parallelize tree sampling.

### 3.3 Calculating Expected Rule Counts

For every observation \( x_d \), the hybrid approach produces a set of sampled trees, each of which contains three types of productions: adapted rules, original PCFG rules, and potentially adapted rules. The last set is most important, as these are new rules discovered by the sampler. These are explained using the Chinese restaurant metaphor in Figure 1. The multiset of all adapted productions is \( M(t_{d,i}) \) and the multiset of non-adapted productions that generate tree \( t_{d,i} \) is \( N(t_{d,i}) \). We compute three counts:

1. \( f \) is the expected number of productions within the TNG. It is the sum over the probability of a tree \( t_{d,k} \) times the number of times an adapted production appeared in \( t_{d,k} \),

\[
g_d(a \rightarrow \beta) = \sum_k \left( \phi_{d,k} \mid a \Rightarrow \alpha : a \Rightarrow \alpha, \beta \in M(t_{d,k}) \right).
\]

2. \( g \) is the expected counts of PCFG productions \( R \) that defines the base distribution of the adaptor grammar, \( g_d(a \rightarrow \beta) = \sum_k \left( \phi_{d,k} \mid a \Rightarrow \beta : a \Rightarrow \beta \in N(t_{d,k}) \right) \).

3. Finally, a third set of productions are newly discovered by the sampler and not in the TNG. These subtrees—lists of PCFG rules sampled from Equation 6—correspond to adapted productions not yet present in the TNG.

### 3.4 Variational Updates

Given the sparse vectors \( \phi \) sampled from the hybrid MCMC step, we update all variational parameters as

\[
\begin{align*}
g_{a \rightarrow \beta} & = \alpha_{a \rightarrow \beta} + \sum_{x_d \in X} g_d(a \rightarrow \beta) \sum_{b \in M} \sum_{i=1}^{K_b} n(a \Rightarrow \beta, z_{a,i}) \\
\mu_{a,i}^1 & = 1 - b_a + \sum_{x_d \in X} f_d(a \Rightarrow \alpha) \sum_{b \in M} \sum_{i=1}^{K_b} n(a \Rightarrow \alpha, z_{a,i}) \\
\mu_{a,i}^2 & = a_a + b_a \sum_{x_d \in X} f_d(a \Rightarrow \alpha) \sum_{b \in M} \sum_{i=1}^{K_b} n(a \Rightarrow \alpha, z_{a,j}, z_{b,k})
\end{align*}
\]

where \( n(r, t) \) is the expected number of times production \( r \) is in tree \( t \), estimated during sampling.

### Hyperparameter Update

We update our PCFG hyperparameter \( \alpha \), PYGEM hyperparameters \( a \) and \( b \) as in Cohen et al. (2010).

### 4 Online Variational Inference

Online inference for probabilistic models requires us to update our posterior distribution as new observations arrive. Unlike batch inference algorithms, we do not assume we have access to the entire...
dataset. Instead, we assume that observations arrive in small groups called minibatches. The advantage of online inference is threefold: a) it does not require retaining the whole dataset in memory; b) each online update is fast; and c) the model usually converges faster. All of these make adaptor grammars scalable to larger datasets.

Our approach is based on the stochastic variational inference for topic models (Hoffman et al., 2013). This inference strategy uses a form of stochastic gradient descent (Bottou, 1998): using the gradient of the ELBO, it finds the sufficient statistics necessary to update variational parameters (which are mostly expected counts calculated using the inside-outside algorithm), and interpolates the result with the current model.

We assume data arrive in minibatches \( B \) (a set of sentences). We accumulate expected counts

\[
\hat{f}^{(l)}(a \Rightarrow z_{a,i}) = (1 - \epsilon) \cdot \hat{f}^{(l-1)}(a \Rightarrow z_{a,i}) + \epsilon \cdot |B| \sum_{d \in B} f_d(a \Rightarrow z_{a,i}),
\]

with decay factor \( \epsilon \in (0, 1) \) to guarantee convergence. We set it to \( \epsilon = (\tau + l)^{-\kappa} \), where \( l \) is the minibatch counter. The decay inertia \( \tau \) prevents premature convergence, and decay rate \( \kappa \) controls the speed of change in sufficient statistics (Hoffman et al., 2010). We recover batch variational approach when \( B = D \) and \( \kappa = 0 \).

The variables \( \hat{f}^{(l)} \) and \( \hat{g}^{(l)} \) are accumulated sufficient statistics of adapted and unadapted productions after processing minibatch \( B_l \). They update the approximate gradient. The updates for variational parameters become

\[
\gamma_{a \Rightarrow \beta} = \alpha_{a \Rightarrow \beta} + \hat{g}^{(l)}(a \Rightarrow \beta)
\]

\[
\nu_{a,i}^1 = 1 - b_a + \hat{f}^{(l)}(a \Rightarrow z_{a,i})
\]

\[
\nu_{a,i}^2 = a_a + i b_a + \sum_{K_a} \hat{f}^{(l)}(a \Rightarrow z_{a,j})
\]

where \( K_a \) is the size of the TNG at adaptor \( a \in M \).

### 4.1 Refining the Truncation

As we observe more data during inference, our TNGs need to change. New rules should be added, useless rules should be removed, and derivations for existing rules should be updated. In this section, we describe heuristics for performing each of these operations.

#### Adding Productions

Sampling can identify productions that are not adapted but were instead drawn from the base distribution. These are candidates for the TNG. For every nonterminal \( a \), we add these potentially adapted productions to TNG\(_a\) after each minibatch. The count associated with candidate productions is now associated with an adapted production, i.e., the \( h \) count contributes to the relevant \( f \) count. This mechanism dynamically expands TNG\(_a\).

#### Sorting and Removing Productions

Our model does not require a preprocessing step to initialize the TNGs, rather, it constructs and expands all TNGs on the fly. To prevent the TNG from growing unwieldy, we prune TNG after every \( n \) minibatches. As a result, we need to impose an ordering over all the parse trees in the TNG. The underlying PYGEM distribution implicitly places a ranking over all the atoms according to their corresponding sufficient statistics (Kurihara et al., 2007), as shown in Equation 9. It measures the “usefulness” of every adapted production throughout inference process.

In addition to accumulated sufficient statistics, Cohen et al. (2010) add a secondary term to discourage short constituents (Mochihashi et al., 2009). We impose a reward term for longer phrases in addition to \( \hat{f} \) and sort all adapted productions in TNG\(_a\) using the ranking score

\[
\Lambda(a \Rightarrow z_{a,i}) = \hat{f}^{(l)}(a \Rightarrow z_{a,i}) \cdot \log(\epsilon \cdot |s| + 1),
\]

where \( |s| \) is the number of yields in production \( a \Rightarrow z_{a,i} \). Because \( \epsilon \) decreases each minibatch, the reward for long phrases diminishes. This is similar to an annealed version of Cohen et al. (2010)—where the reward for long phrases is fixed, see also Mochihashi et al. (2009). After sorting, we remove all but the top \( K_a \) adapted productions.

#### Rederiving Adapted Productions

For MCMC inference, Johnson and Goldwater (2009) observe that atoms already associated with a yield may have trees
Algorithm 2 Online inference for adaptor grammars

1: Random initialize all variational parameters.
2: for minibatch of \( l = 1, 2, \ldots \) do
3: Construct approximate PCFG \( \mathcal{A} \) of \( \mathcal{A} \) (Equation 6).
4: for input sentence \( d = 1, 2, \ldots, D \) do
5: Accumulate inside probabilities from approximate PCFG \( \mathcal{A} \).
6: Sample phrase-structure trees \( \sigma \) and update the tree distribution \( \phi \) (Equation 5).
7: For every adapted nonterminal \( c \), append adapted productions to TNGs.
8: Accumulate sufficient statistics (Equations 9 and 10).
9: Update \( \gamma, \nu^1 \), and \( \nu^2 \) (Equations 11-13).
10: Refine and prune the truncation every \( u \) minibatches.

that do not explain their yield well. They propose table label resampling to rederive yields.

In our approach this is equivalent to "mutating" some derivations in a TNG. After pruning rules every \( u \) minibatches, we perform table label resampling for adapted nonterminals from general to specific (i.e., a topological sort). This provides better expected counts \( n(r, \cdot) \) for rules used in phrase-structure subtrees. Empirically, we find table label resampling only marginally improves the word-segmentation result.

Initialization Our inference begins with random variational Dirichlets and empty TNGs, which obviates the preprocessing step in Cohen et al. (2010). Our model constructs and expands all TNGs on the fly. It mimics the incremental initialization of Johnson and Goldwater (2009). Algorithm 2 summarizes the pseudo-code of our online approach.

4.2 Complexity

Inside and outside calls dominate execution time for adaptor grammar inference. Variational approaches compute inside-outside algorithms and estimate the expected counts for every possible tree derivation (Cohen et al., 2010). For a dataset with \( D \) observations, variational inference requires \( O(DI) \) calls to inside-outside algorithm, where \( I \) is the number of iterations, typically in the tens.

In contrast, MCMC only needs to accumulate inside probabilities, and then sample a tree derivation (Chappelier and Rajman, 2000). The sampling step is negligible in processing time compared to the inside algorithm. MCMC inference requires \( O(D) \) calls to the inside algorithm—hence every iteration is much faster than variational approach—but \( I \) is usually on the order of thousands.

Likewise, our hybrid approach also only needs the less expensive inside algorithm to sample trees. And while each iteration is less expensive, our approach can achieve reasonable results with only a single pass through the data. And thus only requires \( O(D) \) calls to the inside algorithm.

Because the inside-outside algorithm is fundamental to each of these algorithms, we use it as a common basis for comparison across different implementations. This is over-generous to variational approaches, as the full inside-outside computation is more expensive than the inside probability computation required for sampling in MCMC and our hybrid approach.

5 Experiments and Discussion

We implement our online adaptor grammar model (ONLINE) in Python\(^4\) and compare it against both MCMC (Johnson and Goldwater, 2009, MCMC) and the variational inference (Cohen et al., 2010, VARIATIONAL). We use the latest implementation of MCMC sampler for adaptor grammars\(^5\) and simulate the variational approach using our implementation. For MCMC approach, we use the best settings reported in Johnson and Goldwater (2009) with incremental initialization and table label resampling.

\(^4\)Available at http://www.umiacs.umd.edu/~zhaike/.

Table 1: Grammars used in our experiments. The nonterminal CHAR is a non-adapted rule that expands to all characters used in the data, sometimes called pre-terminals. Adapted nonterminals are underlined. For the unigram grammar, only nonterminal WORD is adapted; whereas for the collocation grammar, both nonterminals WORD and COLLOC are adapted. For the infVoc LDA grammar, \( D \) is the total number of documents and \( K \) is the number of topics. Therefore, \( j \) ranges over \( \{1, \ldots, D\} \) and \( i \) ranges over \( \{1, \ldots, K\} \).
Table 2: Word segmentation accuracy measured by word token F1 scores and negative log-likelihood on held-out test dataset in the brackets (lower the better, on the scale of 10^9) for our ONLINE model against MCMC approach (Johnson et al., 2006) on various dataset using the unigram and collocation grammar.\footnote{Our ONLINE settings are batch size $B = 20$, decay inertia $\tau = 128$, decay rate $\kappa = 0.6$ for unigram grammar; and minibatch size $B = 5$, decay inertia $\tau = 256$, decay rate $\kappa = 0.8$ for collocation grammar. TNGs are refined at interval $u = 50$. Truncation size is set to $K_{Word} = 1.5k$ and $K_{Colloc} = 3k$. The settings are chosen from cross validation. We observe similar behavior under $\kappa = \{0.7, 0.9, 1.0\}$, $\tau = \{32, 64, 512\}$, $B = \{10, 50\}$ and $u = \{10, 20, 100\}$.}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline
\textbf{Model and Settings} & \multicolumn{2}{|c|}{\textbf{ctb7}} & \multicolumn{2}{|c|}{\textbf{pku}} & \multicolumn{2}{|c|}{\textbf{cityu}} \\
\hline
\textbf{MCMC} & \textbf{unigram} & \textbf{collocation} & \textbf{unigram} & \textbf{collocation} & \textbf{unigram} & \textbf{collocation} \\
\hline
500 iter & 72.70 (2.81) & 50.53 (2.82) & 72.01 (2.82) & 49.06 (2.81) & 74.19 (3.56) & 63.14 (3.53) \\
1000 iter & 72.65 (2.83) & 62.27 (2.79) & 71.81 (2.81) & 62.47 (2.77) & 74.37 (3.54) & 70.62 (3.51) \\
1500 iter & 72.17 (2.80) & 69.65 (2.77) & 71.46 (2.80) & 70.20 (2.73) & 74.22 (3.54) & 72.33 (3.50) \\
2000 iter & 71.75 (2.79) & 71.66 (2.76) & 71.04 (2.79) & 72.55 (2.70) & 74.01 (3.53) & 73.15 (3.48) \\
\hline
\textbf{ONLINE} & \textbf{unigram} & \textbf{collocation} & \textbf{unigram} & \textbf{collocation} & \textbf{unigram} & \textbf{collocation} \\
\hline
$\kappa$ & \textbf{K\textsubscript{Word}} = 40k & \textbf{K\textsubscript{Colloc}} = 120k & \textbf{K\textsubscript{Word}} = 40k & \textbf{K\textsubscript{Colloc}} = 120k & \textbf{K\textsubscript{Word}} = 40k & \textbf{K\textsubscript{Colloc}} = 150k \\
\hline
0.6 & 32 & 70.17 (2.84) & 68.43 (2.77) & 69.93 (2.89) & 68.09 (2.71) & 72.59 (3.62) & 69.27 (3.61) \\
128 & 72.98 (2.72) & 65.20 (2.81) & 72.26 (2.63) & 65.57 (2.83) & 74.73 (3.40) & 64.83 (3.62) \\
512 & 72.76 (2.78) & 56.05 (2.85) & 71.99 (2.74) & 58.94 (2.94) & 73.68 (3.60) & 60.40 (3.70) \\
\hline
0.8 & 32 & 71.10 (2.77) & 70.84 (2.76) & 70.31 (2.78) & 70.91 (2.71) & 73.12 (3.60) & 71.89 (3.50) \\
128 & 72.79 (2.64) & 70.93 (2.63) & 72.08 (2.62) & 72.02 (2.63) & 74.62 (3.45) & 72.28 (3.51) \\
512 & 72.82 (2.58) & 68.53 (2.76) & 72.14 (2.58) & 70.07 (2.90) & 74.71 (3.37) & 72.58 (3.49) \\
\hline
1.0 & 32 & 69.98 (2.87) & 70.71 (2.63) & 69.42 (2.84) & 71.45 (2.67) & 74.18 (3.59) & 72.42 (3.45) \\
128 & 71.84 (2.72) & 71.29 (2.58) & 71.29 (2.67) & 72.56 (2.61) & 73.23 (3.39) & 72.61 (3.41) \\
512 & 72.68 (2.62) & 70.67 (2.60) & 71.86 (2.63) & 71.39 (2.66) & 74.45 (3.41) & 72.88 (3.38) \\
\hline
\textbf{VARIATIONAL} & \textbf{unigram} & \textbf{collocation} & \textbf{unigram} & \textbf{collocation} & \textbf{unigram} & \textbf{collocation} \\
\hline
60.83 (2.85) & 67.78 (2.75) & 67.82 (2.80) & 66.97 (2.75) & 70.47 (3.72) & 69.06 (3.69) \\
\hline
\end{tabular}
\end{table}

\subsection{5.1 Word Segmentation}

We evaluate our online adaptor grammar on the task of word segmentation, which focuses on identifying word boundaries from a sequence of characters. This is especially the case for Chinese, since characters are written in sequence without word boundaries.

We first evaluate all three models on the standard Brent version of the Bernstein-Ratner corpus (Bernstein-Ratner, 1987; Brent and Cartwright, 1996, brent). The dataset contains 10k sentences, 1.3k distinct words, and 72 distinct characters. We compare the results on both unigram and collocation grammars introduced in Johnson and Goldwater (2009) as listed in Table 1.

Figure 2 illustrates the word segmentation accuracy in terms of word token F1 scores on brent against the number of inside-outside function calls for all three approaches using unigram and collocation grammars. In both cases, our ONLINE approach converges faster than MCMC and VARIATIONAL approaches, yet yields comparable or better performance when seeing more data.

In addition to the brent corpus, we also evaluate three approaches on three other Chinese datasets compiled by Xue et al. (2005) and Emerson (2005).\footnote{We use all punctuation as natural delimiters (i.e., words cannot cross punctuation).}

- Chinese Treebank 7.0 (ctb7): 162k sentences, 57k distinct words, 4.5k distinct characters; our model under $\kappa = \{0.7, 0.9\}$ and $\tau = \{64, 256\}$.}
- Peking University (pku): 183k sentences, 53k distinct words, 4.6k distinct characters; and
- City University of Hong Kong (cityu): 207k sentences, 64k distinct words, and 5k distinct characters.

We compare our inference method against other approaches on F1 score. While other unsupervised word segmentation systems are available (Mochihashi et al. (2009), inter alia), our focus is on a direct comparison of inference techniques for adaptor grammar, which achieve competitive (if not state-of-the-art) performance.

Table 2 shows the word token F1-scores and negative likelihood on held-out test dataset of our model against MCMC and VARIATIONAL. We randomly sample 30% of the data for testing and the rest for training. We compute the held-out likelihood of the most likely sampled parse trees out of each model. Our ONLINE approach consistently better segments words than VARIATIONAL and achieves comparable or better results than MCMC.

For MCMC, Johnson and Goldwater (2009) show that incremental initialization—or online updates in general—results in more accurate word segmentation, even though the trees have lower posterior probability. Similarly, our ONLINE approach initializes and learns them on the fly, instead of initializing the grammatons and parse trees for all data upfront as for VARIATIONAL. This uniformly outperforms batch initialization on the word segmentation tasks.

### 5.2 Infinite Vocabulary Topic Modeling

Topic models often can be replicated using a carefully crafted PCFG (Johnson, 2010). These powerful extensions can capture topical collocations and sticky topics; these embellishments could further improve NLP applications of simple unigram topic models such as word sense disambiguation (Boyd-Graber and Blei, 2007), part of speech tagging (Toutanova and Johnson, 2008) or dialogue modeling (Zhai and Williams, 2014). However, expressing topic models in adaptor grammars is much slower than traditional topic models, for which fast online inference (Hoffman et al., 2010) is available.

Zhai and Boyd-Graber (2013) argue that online inference and topic models violate a fundamental assumption in online algorithms: new words are introduced as more data are streamed to the algorithm. Zhai and Boyd-Graber (2013) introduce an inference framework, InfVOC, to discover words from a Dirichlet process with a character n-gram base distribution.

We show that their complicated model and online inference can be captured and extended via an appropriate PCFG grammar and our online adaptor grammar inference algorithm. Our extension to InfVOC generalizes their static character n-gram model, learning the base distribution (i.e., how words are composed from characters) from data. In contrast, their base distribution was learned from a dictionary as a preprocessing step and held fixed.

This is an attractive testbed for our online inference. Within a topic, we can verify that the words we discover are relevant to the topic and that new words rise in importance in the topic over time if they are relevant. For these experiments, we treat each token (with its associated document pseudo-word \(\hat{\ell}_j\)) as a single sentence, and each minibatch contains only one sentence (token).
Quantitatively, we evaluate three different inference schemes and the InfVoc approach\(^{13}\) on a collection of English daily news snippets (de-news).\(^ {14}\) We used the InfVoc LDA grammar (Table 1). For all approaches, we train the model with five topics, and evaluate topic coherence (Newman et al., 2009), which correlates well with human ratings of topic interpretability (Chang et al., 2009). We collect the co-occurrence counts from Wikipedia and compute the average pairwise pointwise mutual information (PMI) score between the top 10 ranked words of every topic. Figure 3 illustrates the PMI score for both approaches. Our approach yields comparable or better results against all other approaches under most conditions.

Qualitatively, Figure 4 shows an example of a topic evolution using online adaptor grammar for the de-news dataset. The topic is about “tax policy”. The topic improves over time; words like “year”, “tax” and “minist(er)” become more prominent. More importantly, the online approach discovers new words and incorporates them into the topic. For example, “schroeder” (former German chancellor) first appeared in minibatch 300, was successfully picked up by our model, and became one of the top ranked words in the topic.

6 Conclusion

Probabilistic modeling is a useful tool in understanding unstructured data or data where the structure is latent, like language. However, developing these models is often a difficult process, requiring significant machine learning expertise.

Adaptor grammars offer a flexible and quick way to prototype and test new models. Despite expensive inference, they have been used for topic modeling (Johnson, 2010), discovering perspective (Hardisty et al., 2010), segmentation (Johnson and Goldwater, 2009), and grammar induction (Cohen et al., 2010).

We have presented a new online, hybrid inference scheme for adaptor grammars. Unlike previous approaches, it does not require extensive preprocessing. It is also able to faster discover useful structure in text; with further development, these algorithms could further speed the development and application of new nonparametric models to large datasets.

---

\(^{13}\)Available at [http://www.umiacs.umd.edu/~zhaike/](http://www.umiacs.umd.edu/~zhaike/).

\(^{14}\)The de-news dataset is randomly selected subset of 2.2k English documents from [http://homepages.inf.ed.ac.uk/pkoehn/publications/de-news/](http://homepages.inf.ed.ac.uk/pkoehn/publications/de-news/). It contains 6.5k unique types and over 200k word tokens. Tokenization and stemming provided by NLTK (Bird et al., 2009).
Acknowledgments

We would like to thank the anonymous reviewers, Kristina Toutanova, Mark Johnson, and Ke Wu for insightful discussions. This work was supported by NSF Grant CCF-1018625. Boyd-Graber is also supported by NSF Grant IIS-1320538. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsor.

References


Michael R. Brent and Timothy A. Cartwright. 1996. Distributional regularity and phonotactic constraints are useful for segmentation. volume 61, pages 93–125.


