Improving nonparameteric Bayesian inference: experiments on unsupervised word segmentation with adaptor grammars

Mark Johnson
Brown University
Providence, RI
Mark_Johnson@Brown.edu

Sharon Goldwater
University of Edinburgh
Edinburgh EH8 9AB
sgwater@inf.ed.ac.uk

Abstract

One of the reasons nonparametric Bayesian inference is attracting attention in computational linguistics is because it provides a principled way of learning the units of generalization together with their probabilities. Adaptor grammars are a framework for defining a variety of hierarchical nonparametric Bayesian models. This paper investigates some of the choices that arise in formulating adaptor grammars and associated inference procedures, and shows that they can have a dramatic impact on performance in an unsupervised word segmentation task. With appropriate adaptor grammars and inference procedures we achieve an 87% word token f-score on the standard Brent version of the Bernstein-Ratner corpus, which is an error reduction of over 35% over the best previously reported results for this corpus.

1 Introduction

Most machine learning algorithms used in computational linguistics are *parametric*, i.e., they learn a numerical weight (e.g., a probability) associated with each feature, where the set of features is fixed before learning begins. Such procedures can be used to learn features or structural units by embedding them in a "propose-and-prune" algorithm: a feature proposal component proposes potentially useful features (e.g., combinations of the currently most useful features), which are then fed to a parametric learner that estimates their weights. After estimating feature weights and pruning "useless" low-weight features, the cycle repeats. While such algorithms can achieve impressive results (Stolcke and Omohundro, 1994), their effectiveness depends on how well the feature proposal step relates to the overall learning objective, and it can take considerable insight and experimentation to devise good feature proposals.

One of the main reasons for the recent interest in *nonparametric Bayesian inference* is that it offers a systematic framework for structural inference, i.e., inferring the features relevant to a particular problem as well as their weights. (Here "nonparametric" means that the models do not have a fixed set of parameters; our nonparametric models do have parameters, but the particular parameters in a model are learned along with their values). Dirichlet Processes and their associated predictive distributions, Chinese Restaurant Processes, are one kind of nonparametric Bayesian model that has received considerable attention recently, in part because they can be composed in hierarchical fashion to form Hierarchical Dirichlet Processes (HDP) (Teh et al., 2006).

Lexical acquisition is an ideal test-bed for exploring methods for inferring structure, where the features learned are the words of the language. (Even the most hard-core nativists agree that the words of a language must be learned). We use the unsupervised word segmentation problem as a test case for evaluating structural inference in this paper. Nonparametric Bayesian methods produce state-of-the-art performance on this task (Goldwater et al., 2006a; Goldwater et al., 2007; Johnson, 2008).

In a computational linguistics setting it is natural to try to align the HDP hierarchy with the hierarchy defined by a grammar. Adaptor grammars, which are one way of doing this, make it easy to explore a wide variety of HDP grammar-based models. Given an appropriate adaptor grammar, the fea-

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 317–325,
Boulder, Colorado, June 2009.

tures learned by adaptor grammars can correspond to linguistic units such as words, syllables and collocations. Different adaptor grammars encode different assumptions about the structure of these units and how they relate to each other. A generic adaptor grammar inference program infers these units from training data, making it easy to investigate how these assumptions affect learning (Johnson, 2008).[1]

However, there are a number of choices in the design of adaptor grammars and the associated inference procedure. While this paper studies the impact of these on the word segmentation task, these choices arise in other nonparametric Bayesian inference problems as well, so our results should be useful more generally. The rest of this paper is organized as follows. The next section reviews adaptor grammars and presents three different adaptor grammars for word segmentation that serve as running examples in this paper. Adaptor grammars contain a large number of adjustable parameters, and Section 3 discusses how these can be estimated using Bayesian techniques. Section 4 examines several implementation options within the adaptor grammar inference algorithm and shows that they can make a significant impact on performance. Cumulatively these changes make a significant difference in word segmentation accuracy: our final adaptor grammar performs unsupervised word segmentation with an 87% token f-score on the standard Brent version of the Bernstein-Ratner corpus (Bernstein-Ratner, 1987; Brent and Cartwright, 1996), which is an error reduction of over 35% compared to the best previously reported results on this corpus.

2 Adaptor grammars

This section informally introduces adaptor grammars using unsupervised word segmentation as a motivating application; see Johnson et al. (2007b) for a formal definition of adaptor grammars.

Consider the problem of learning language from continuous speech: segmenting each utterance into words is a nontrivial problem that language learners must solve. Elman (1990) introduced an idealized version of this task, and Brent and Cartwright (1996) presented a version of it where the data consists of unsegmented phonemic representations of the sentences in the Bernstein-Ratner corpus of child-directed speech (Bernstein-Ratner, 1987). Because these phonemic representations are obtained by looking up orthographic forms in a pronouncing dictionary and appending the results, identifying the word tokens is equivalent to finding the locations of the word boundaries. For example, the phoneme string corresponding to "you want to see the book" (with its correct segmentation indicated) is as follows:

$$\mathrm{y}\ \vartriangle\ \mathrm{u}\ \blacktriangle\ \mathrm{w}\ \vartriangle\ \mathrm{a}\ \vartriangle\ \mathrm{n}\ \vartriangle\ \mathrm{t}\ \blacktriangle\ \mathrm{t}\ \vartriangle\ \mathrm{u}\ \blacktriangle\ \mathrm{s}\ \vartriangle\ \mathrm{i}\ \blacktriangle\ \mathrm{D}\ \vartriangle\ 6\ \blacktriangle\ \mathrm{b}\ \vartriangle\ \mathrm{U}\ \vartriangle\ \mathrm{k}$$

We can represent any possible segmentation of any possible sentence as a tree generated by the following *unigram grammar*.

$$\begin{aligned}\mathrm{Sentence} &\rightarrow \underline{\mathrm{Word}}^{+}\\ \underline{\mathrm{Word}} &\rightarrow \mathrm{Phoneme}^{+}\end{aligned}$$

The nonterminal Phoneme expands to each possible phoneme; the underlining, which identifies "adapted nonterminals", will be explained below. In this paper "+" abbreviates right-recursion through a dummy nonterminal, i.e., the unigram grammar actually is:

$$\begin{aligned}\mathrm{Sentence} &\rightarrow \underline{\mathrm{Word}}\\ \mathrm{Sentence} &\rightarrow \underline{\mathrm{Word}}\ \mathrm{Sentence}\\ \underline{\mathrm{Word}} &\rightarrow \mathrm{Phonemes}\\ \mathrm{Phonemes} &\rightarrow \mathrm{Phoneme}\\ \mathrm{Phonemes} &\rightarrow \mathrm{Phoneme}\ \mathrm{Phonemes}\end{aligned}$$

A PCFG with these productions can represent all possible segmentations of any Sentence into a sequence of Words. But because it assumes that the probability of a word is determined purely by multiplying together the probability of its individual phonemes, it has no way to encode the fact that certain strings of phonemes (the words of the language) have much higher probabilities than other strings containing the same phonemes. In order to do this, a PCFG would need productions like the following one, which encodes the fact that "want" is a Word.

$$\mathrm{Word} \rightarrow \mathrm{w}\ \mathrm{a}\ \mathrm{n}\ \mathrm{t}$$

Adaptor grammars can be viewed as a way of formalizing this idea. Adaptor grammars learn the probabilities of entire subtrees, much as in tree substitution grammar (Joshi, 2003) and DOP (Bod,

[1]The adaptor grammar inference program is available for download at http://www.cog.brown.edu/~mj/Software.htm.

1998). (For computational efficiency reasons adaptor grammars require these subtrees to expand to terminals). The set of possible adapted tree fragments is the set of all subtrees generated by the CFG whose root label is a member of the set of *adapted nonterminals* A (adapted nonterminals are indicated by underlining in this paper). For example, in the unigram adaptor grammar $A = \{\mathrm{Word}\}$, which means that the adaptor grammar inference procedure learns the probability of each possible Word subtree. Thus adaptor grammars are simple models of structure learning in which adapted subtrees are the units of generalization.

One might try to reduce adaptor grammar inference to PCFG parameter estimation by introducing a context-free rule for each possible adapted subtree, but such an attempt would fail because the number of such adapted subtrees, and hence the number of corresponding rules, is unbounded. However nonparametric Bayesian inference techniques permit us to *sample* from this infinite set of adapted subtrees, and only require us to instantiate the finite number of them needed to analyse the finite training data.

An *adaptor grammar* is a 7-tuple $(N, W, R, S, \boldsymbol{\theta}, A, \boldsymbol{C})$ where $(N, W, R, S, \boldsymbol{\theta})$ is a PCFG with nonterminals N, terminals W, rules R, start symbol $S \in N$ and rule probabilities $\boldsymbol{\theta}$, where θ_r is the probability of rule $r \in R$, $A \subseteq N$ is the set of *adapted nonterminals* and $\boldsymbol{C}$ is a vector of *adaptors* indexed by elements of A, so C_X is the adaptor for adapted nonterminal $X \in A$.

Informally, an adaptor C_X nondeterministically maps a stream of trees from a *base distribution* H_X whose support is $\mathcal{T}_X$ (the set of subtrees whose root node is $X \in N$ generated by the grammar's rules) into another stream of trees whose support is also $\mathcal{T}_X$. In adaptor grammars the base distributions H_X are determined by the PCFG rules expanding X and the other adapted distributions, as explained in Johnson et al. (2007b). When called upon to generate another sample tree, the adaptor either generates and returns a fresh tree from H_X or regenerates a tree it has previously emitted, so in general the adapted distribution differs from the base distribution.

This paper uses adaptors based on Chinese Restaurant Processes (CRPs) or Pitman-Yor Processes (PYPs) (Pitman, 1995; Pitman and Yor, 1997; Ishwaran and James, 2003). CRPs and PYPs nondeterministically generate infinite sequences of natural numbers $z_1, z_2, \ldots$, where $z_1 = 1$ and each $z_{n+1} \leq m + 1$ where $m = \max(z_1, \ldots, z_n)$. In the "Chinese Restaurant" metaphor samples produced by the adaptor are viewed as "customers" and z_n is the index of the "table" that the nth customer is seated at. In adaptor grammars each table in the adaptor C_X is labeled with a tree sampled from the base distribution H_X that is shared by all customers at that table; thus the nth sample tree from the adaptor C_X is the z_nth sample from H_X.

CRPs and PYPs differ in exactly how the sequence $\{z_k\}$ is generated. Suppose $\boldsymbol{z} = (z_1, \ldots, z_n)$ have already been generated and $m = \max(\boldsymbol{z})$. Then a CRP generates the next table index z_{n+1} according to the following distribution:

$$\mathrm{P}(Z_{n+1} = k \mid \boldsymbol{z}) \propto \begin{cases} n_k(\boldsymbol{z}) & \text{if } k \leq m \\ \alpha & \text{if } k = m + 1 \end{cases}$$

where $n_k(\boldsymbol{z})$ is the number of times table k appears in $\boldsymbol{z}$ and $\alpha > 0$ is an adjustable parameter that determines how often a new table is chosen. This means that if C_X is a CRP adaptor then the next tree t_{n+1} it generates is the same as a previously generated tree t' with probability proportional to the number of times C_X has generated t' before, and is a "fresh" tree t sampled from H_X with probability proportional to $\alpha_X H_X(t)$. This leads to a powerful "rich-get-richer" effect in which popular trees are generated with increasingly high probabilities.

Pitman-Yor Processes can control the strength of this effect somewhat by moving mass from existing tables to the base distribution. The PYP predictive distribution is:

$$\mathrm{P}(Z_{n+1} = k \mid \boldsymbol{z}) \propto \begin{cases} n_k(\boldsymbol{z}) - a & \text{if } k \leq m \\ m\,a + b & \text{if } k = m + 1 \end{cases}$$

where $a \in [0, 1]$ and $b > 0$ are adjustable parameters. It's easy to see that the CRP is a special case of the PRP where $a = 0$ and $b = \alpha$.

Each adaptor in an adaptor grammar can be viewed as estimating the probability of each adapted subtree t; this probability can differ substantially from t's probability $H_X(t)$ under the base distribution. Because Words are adapted in the unigram adaptor grammar it effectively estimates the probability of each Word tree separately; the sampling estimators described in section 4 only instantiate those Words actually used in the analysis of $\mathrm{Sentences}$ in the corpus. While the Word adaptor will generally

prefer to reuse Words that have been used elsewhere in the corpus, it is always possible to generate a fresh Word using the CFG rules expanding Word into a string of Phonemes.

We assume for now that all CFG rules R_X expanding the nonterminal $X \in N$ have the same probability (although we will explore estimating $\boldsymbol{\theta}$ below), so the base distribution H_{Word} is a "monkeys banging on typewriters" model. That means the unigram adaptor grammar implements the Goldwater et al. (2006a) unigram word segmentation model, and in fact it produces segmentations of similar accuracies, and exhibits the same characteristic undersegmentation errors. As Goldwater et al. point out, because Words are the only units of generalization available to a unigram model it tends to misanalyse collocations as words, resulting in a marked tendancy to undersegment.

Goldwater et al. demonstrate that modelling bigram dependencies mitigates this undersegmentation. While adaptor grammars cannot express the Goldwater et al. bigram model, they can get much the same effect by directly modelling collocations (Johnson, 2008). A *collocation adaptor grammar* generates a Sentence as a sequence of Collocations, each of which expands to a sequence of Words.

$$\begin{aligned}\text{Sentence} &\rightarrow \underline{\text{Colloc}}^+\\ \underline{\text{Colloc}} &\rightarrow \underline{\text{Word}}^+\\ \underline{\text{Word}} &\rightarrow \text{Phoneme}^+\end{aligned}$$

Because Colloc is adapted, the collocation adaptor grammar learns Collocations as well as Words. (Presumably these approximate syntactic, semantic and pragmatic interword dependencies). Johnson reported that the collocation adaptor grammar segments as well as the Goldwater et al. bigram model, which we confirm here.

Recently other researchers have emphasised the utility of phonotactic constraints (i.e., modeling the allowable phoneme sequences at word onsets and endings) for word segmentation (Blanchard and Heinz, 2008; Fleck, 2008). Johnson (2008) points out that adaptor grammars that model words as sequences of syllables can learn and exploit these constraints, significantly improving segmentation accuracy. Here we present an adaptor grammar that models collocations together with these phonotactic constraints. This grammar is quite complex, permitting us to study the effects of the various model and implementation choices described below on a complex hierarchical nonparametric Bayesian model.

The *collocation-syllable adaptor grammar* generates a Sentence in terms of three levels of Collocations (enabling it to capture a wider range of interword dependencies), and generates Words as sequences of 1 to 4 Syllables. Syllables are subcategorized as to whether they are initial (I), final (F) or both (IF).

$$\begin{aligned}\text{Sentence} &\rightarrow \underline{\text{Colloc3}}^+\\ \underline{\text{Colloc3}} &\rightarrow \underline{\text{Colloc2}}^+\\ \underline{\text{Colloc2}} &\rightarrow \underline{\text{Colloc1}}^+\\ \underline{\text{Colloc1}} &\rightarrow \underline{\text{Word}}^+\\ \underline{\text{Word}} &\rightarrow \text{SyllableIF}\\ \underline{\text{Word}} &\rightarrow \text{SyllableI (Syllable) (Syllable) SyllableF}\\ \text{Syllable} &\rightarrow \underline{\text{Onset}}\ \text{Rhyme}\\ \underline{\text{Onset}} &\rightarrow \text{Consonant}^+\\ \text{Rhyme} &\rightarrow \underline{\text{Nucleus}}\ \underline{\text{Coda}}\\ \underline{\text{Nucleus}} &\rightarrow \text{Vowel}^+\\ \underline{\text{Coda}} &\rightarrow \text{Consonant}^+\\ \text{SyllableIF} &\rightarrow \underline{\text{OnsetI}}\ \text{RhymeF}\\ \underline{\text{OnsetI}} &\rightarrow \text{Consonant}^+\\ \text{RhymeF} &\rightarrow \underline{\text{Nucleus}}\ \underline{\text{CodaF}}\\ \underline{\text{CodaF}} &\rightarrow \text{Consonant}^+\\ \text{SyllableI} &\rightarrow \underline{\text{OnsetI}}\ \text{Rhyme}\\ \text{SyllableF} &\rightarrow \underline{\text{Onset}}\ \text{RhymeF}\end{aligned}$$

Here Consonant and Vowel expand to all possible consonants and vowels respectively, and the parentheses in the expansion of Word indicate optionality. Because Onsets and Codas are adapted, the collocation-syllable adaptor grammar learns the possible consonant sequences that begin and end syllables. Moreover, because Onsets and Codas are subcategorized based on whether they are word-peripheral, the adaptor grammar learns which consonant clusters typically appear at word boundaries, even though the input contains no explicit word boundary information (apart from what it can glean from the sentence boundaries).

3 Bayesian estimation of adaptor grammar parameters

Adaptor grammars as defined in section 2 have a large number of free parameters that have to be chosen by the grammar designer; a rule probability θ_r for each PCFG rule $r \in R$ and either one or two hyperparameters for each adapted nonterminal $X \in A$, depending on whether Chinese Restaurant

or Pitman-Yor Processes are used as adaptors. It's difficult to have intuitions about the appropriate settings for the latter parameters, and finding the optimal values for these parameters by some kind of exhaustive search is usually computationally impractical. Previous work has adopted an expedient such as parameter tying. For example, Johnson (2008) set $\boldsymbol{\theta}$ by requiring all productions expanding the same nonterminal to have the same probability, and used Chinese Restaurant Process adaptors with tied parameters α_X, which was set using a grid search.

We now describe two methods of dealing with the large number of parameters in these models that are both more principled and more practical than the approaches described above. First, we can integrate out $\boldsymbol{\theta}$, and second, we can infer values for the adaptor hyperparameters using sampling. These methods (the latter in particular) make it practical to use Pitman-Yor Process adaptors in complex grammars such as the collocation-syllable adaptor grammar, where it is impractical to try to find optimal parameter values by grid search. As we will show, they also improve segmentation accuracy, sometimes dramatically.

3.1 Integrating out θ

Johnson et al. (2007a) describe Gibbs samplers for Bayesian inference of PCFG rule probabilities $\boldsymbol{\theta}$, and these techniques can be used directly with adaptor grammars as well. Just as in that paper, we place Dirichlet priors on $\boldsymbol{\theta}$: here $\boldsymbol{\theta}_X$ is the subvector of $\boldsymbol{\theta}$ corresponding to rules expanding nonterminal $X \in N$, and $\boldsymbol{\beta}_X$ is a corresponding vector of positive real numbers specifying the hyperparameters of the corresponding Dirichlet distributions:

$$\mathrm{P}(\boldsymbol{\theta} \mid \boldsymbol{\beta}) \quad = \quad \prod_{X \in N} \mathrm{Dir}(\boldsymbol{\theta}_X \mid \boldsymbol{\beta}_X)$$

Because the Dirichlet distribution is conjugate to the multinomial distribution, it is possible to integrate out the rule probabilities $\boldsymbol{\theta}$, producing the "collapsed sampler" described in Johnson et al. (2007a).

In our experiments we chose an uniform prior $\beta_r = 1$ for all rules $r \in R$. As Table 1 shows, integrating out $\boldsymbol{\theta}$ only has a major effect on results when the adaptor hyperparameters themselves are not sampled, and even then it did not have a large effect on the collocation-syllable adaptor grammar. This is not too surprising: because the Onset, $\mathrm{Nucleus}$ and Coda adaptors in this grammar learn the probabilities of these building blocks of words, the phoneme probabilities (which is most of what $\boldsymbol{\theta}$ encodes) play less important a role.

3.2 Slice sampling adaptor hyperparameters

As far as we know, there are no conjugate priors for the adaptor hyperparameters a_X or b_X (which corresponds to α_X in a Chinese Restaurant Process), so it is not possible to integrate them out as we did with the rule probabilities $\boldsymbol{\theta}$. However, it is possible to perform Bayesian inference by putting a prior on them and sampling their values.

Because we have no strong intuitions about the values of these parameters we chose uninformative priors. We chose a uniform $\mathrm{Beta}(1, 1)$ prior on a_X, and a "vague" $\mathrm{Gamma}(10, 0.1)$ prior on $b_X = \alpha_X$ (MacKay, 2003). (We experimented with other parameters in the Gamma prior, but found no significant difference in performance).

After each Gibbs sweep through the parse trees $\boldsymbol{t}$ we resampled each of the adaptor parameters from the posterior distribution of the parameter using a slice sampler 10 times. For example, we resample each b_X from:

$$\mathrm{P}(b_X \mid \boldsymbol{t}) \quad \propto \quad \mathrm{P}(\boldsymbol{t} \mid b_X)\, \mathrm{Gamma}(b_X \mid 10, 0.1)$$

Here $\mathrm{P}(\boldsymbol{t} \mid b_X)$ is the likelihood of the current sequence of sample parse trees (we only need the factors that depend on b_X) and $\mathrm{Gamma}(b_X \mid 10, 0.1)$ is the prior. The same formula is used for sampling a_X, except that the prior is now a flat $\mathrm{Beta}(1, 1)$ distribution.

In general we cannot even compute the normalizing constants for these posterior distributions, so we chose a sampler that does not require this. We use a slice sampler here because it does not require a proposal distribution (Neal, 2003). (We initially tried a Metropolis-Hastings sampler but were unable to find a proposal distribution that had reasonable acceptance ratios for all of our adaptor grammars).

As Table 1 makes clear, sampling the adaptor parameters makes a significant difference, especially on the collocation-syllable adaptor grammar. This is not surprising, as the adaptors in that grammar play many different roles and there is no reason to to expect the optimal values of their parameters to be similar.

Condition					Word token f-scores					
					Sample average			Max. Marginal		
Batch initialization	Table label resampling	Integrate out $\boldsymbol{\theta}$	Sample $\alpha_X = b_X$	Sample a_X	unigram	colloc	colloc-syll	unigram	colloc	colloc-syll
•	•	•	•	•	0.55	0.74	0.85	0.56	0.76	0.87
•	•	•	•		0.55	0.72	0.84	0.56	0.74	0.84
•	•	•			0.55	0.72	0.78	0.57	0.75	0.78
•	•				0.54	0.66	0.75	0.56	0.69	0.76
•	•		•	•	0.54	0.70	0.87	0.56	0.74	0.88
•		•	•	•	0.55	0.42	0.54	0.57	0.51	0.55
	•	•	•	•	0.74	0.83	0.88	0.81	0.86	0.89
		•	•	•	0.75	0.43	0.74	0.80	0.56	0.82
			•	•	0.71	0.41	0.76	0.77	0.49	0.82
	•		•	•	0.71	0.73	0.87	0.77	0.75	0.88

Table 1: Word segmentation accuracy measured by word token f-scores on Brent's version of the Bernstein-Ratner corpus as a function of adaptor grammar, adaptor and estimation procedure. Pitman-Yor Process adaptors were used when a_X was sampled, otherwise Chinese Restaurant Process adaptors were used. In runs where $\boldsymbol{\theta}$ was not integrated out it was set uniformly, and all $\alpha_X = b_X$ were set to 100 they were not sampled.

4 Inference for adaptor grammars

Johnson et al. (2007b) describe the basic adaptor grammar inference procedure that we use here. That paper leaves unspecified a number of implementation details, which we show can make a crucial difference to segmentation accuracy. The adaptor grammar algorithm is basically a Gibbs sampler of the kind widely used for nonparametric Bayesian inference (Blei et al., 2004; Goldwater et al., 2006b; Goldwater et al., 2006a), so it seems reasonable to expect that at least some of the details discussed below will be relevant to other applications as well.

The inference algorithm maintains a vector $\boldsymbol{t} = (t_1, \ldots, t_n)$ of sample parses, where $t_i \in \mathcal{T}_S$ is a parse for the ith sentence w_i. It repeatedly chooses a sentence w_i at random and resamples the parse tree t_i for w_i from $\mathrm{P}(t_i \mid \boldsymbol{t}_{-i}, w_i)$, i.e., conditioned on w_i and the parses $\boldsymbol{t}_{-i}$ of all sentences *except* w_i.

4.1 Maximum marginal decoding

Sampling algorithms like ours produce a stream of samples from the posterior distribution over parses of the training data. It is standard to take the output of the algorithm to be the last sample produced, and evaluate those parses. In some other applications of nonparametric Bayesian inference involving latent structure (e.g., clustering) it is difficult to usefully exploit multiple samples, but that is not the case here.

In *maximum marginal decoding* we map each sample parse tree t onto its corresponding word segmentation s, marginalizing out irrelevant detail in t. (For example, the collocation-syllable adaptor grammar contains a syllabification and collocational structure that is irrelevant for word segmentation). Given a set of sample parse trees for a sentence we compute the set of corresponding word segmentations, and return the one that occurs most frequently (this is a sampling approximation to the maximum probability marginal structure).

For each setting in the experiments described in Table 1 we ran 8 samplers for 2,000 iterations (i.e., passes through the training data), and kept the sample parse trees from every 10th iteration after iteration 1000, resulting in 800 sample parses for every sentence. (An examination of the posterior probabilities suggests that all of the samplers using batch initialization and table label resampling had "burnt

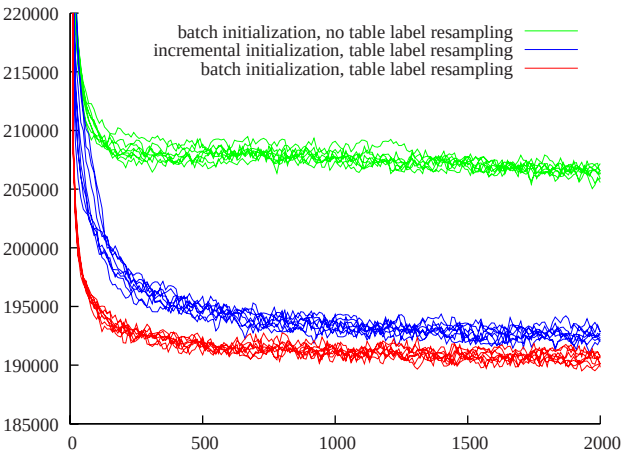

Figure 1: Negative log posterior probability (lower is better) as a function of iteration for 24 runs of the collocation adaptor grammar samplers with Pitman-Yor adaptors. The upper 8 runs use batch initialization but no table label resampling, the middle 8 runs use incremental initialization and table label resampling, while the lower 8 runs use batch initialization and table label resampling.

in" by iteration 1000). We evaluated the word token f-score of the most frequent marginal word segmentation, and compared that to average of the word token f-score for the 800 samples, which is also reported in Table 1. For each grammar and setting we tried, the maximum marginal segmentation was better than the sample average, sometimes by a large margin. Given its simplicity, this suggests that maximum marginal decoding is probably worth trying when applicable.

4.2 Batch initialization

The Gibbs sampling algorithm is initialized with a set of sample parses $\boldsymbol{t}$ for each sentence in the training data. While the fundamental theorem of Markov Chain Monte Carlo guarantees that eventually samples will converge to the posterior distribution, it says nothing about how long the "burn in" phase might last (Robert and Casella, 2004). In practice initialization can make a huge difference to the performance of Gibbs samplers (just as it can with other unsupervised estimation procedures such as Expectation Maximization).

There are many different ways in which we could generate the initial trees $\boldsymbol{t}$; we only study two of the obvious methods here. *Batch initialization* assigns every sentence a random parse tree in parallel. In more detail, the initial parse tree t_i for sentence w_i is sampled from $\mathrm{P}(t \mid w_i, G')$, where G' is the PCFG obtained from the adaptor grammar by ignoring its last two components A and C (i.e., the adapted nonterminals and their adaptors), and seated at a new table. This means that in batch initialization each initial parse tree is randomly generated without any adaptation at all.

Incremental initialization assigns the initial parse trees t_i to sentences w_i in order, updating the adaptor grammar as it goes. That is, t_i is sampled from $\mathrm{P}(t \mid w_i, t_1, \ldots, t_{i-1})$. This is easy to do in the context of Gibbs sampling, since this distribution is a minor variant of the distribution $\mathrm{P}(t_i \mid \boldsymbol{t}_{-i}, w_i)$ used during Gibbs sampling itself.

Incremental initialization is greedier than batch initialization, and produces initial sample trees with much higher probability. As Table 1 shows, across all grammars and conditions after 2,000 iterations incremental initialization produces samples with much better word segmentation token f-score than does batch initialization, with the largest improvement on the unigram adaptor grammar.

However, incremental initialization results in sample parses with lower posterior probability for the unigram and collocation adaptor grammars (but not for the collocation-syllable adaptor grammar). Figure 1 plots the posterior probabilities of the sample trees $\boldsymbol{t}$ at each iteration for the collocation adaptor grammar, showing that even after 2,000 iterations incremental initialization results in trees that are much less likely than those produced by batch initialization. It seems that with incremental initialization the Gibbs sampler gets stuck in a local optimum which it is extremely unlikely to move away from.

It is interesting that incremental initialization results in more accurate word segmentation, even though the trees it produces have lower posterior probability. This seems to be because the most probable analyses produced by the unigram and, to a lesser extent, the collocation adaptor grammars tend to undersegment. Incremental initialization greedily searches for common substrings, and because such substrings are more likely to be short rather than long, it tends to produce analyses with shorter words than batch initialization does. Goldwater et al. (2006a) show that Brent's incremental segmentation algorithm (Brent, 1999) has a similar property.

We favor batch initialization because we are in-

terested in understanding the properties of our models (expressed here as adaptor grammars), and batch initialization does a better job of finding the most probable analyses under these models. However, it might be possible to justify incremental initialization as (say) cognitively more plausible.

4.3 Table label resampling

Unlike the previous two implementation choices which apply to a broad range of algorithms, table label resampling is a specialized kind of Gibbs step for adaptor grammars and similar hierarchical models that is designed to improve mobility. The adaptor grammar algorithm described in Johnson et al. (2007b) repeatedly resamples parses for the *sentences* of the training data. However, the adaptor grammar sampler itself maintains of a hierarchy of Chinese Restaurant Processes or Pitman-Yor Processes, one per adapted nonterminal $X \in A$, that cache subtrees from $\mathcal{T}_X$. In general each of these subtrees will occur many times in the parses for the training data sentences. Table label resampling resamples the trees in these adaptors (i.e., the table labels, to use the restaurant metaphor), potentially changing the analysis of many sentences at once. For example, each $\mathrm{Collocation}$ in the collocation adaptor grammar can occur in many $\mathrm{Sentences}$, and each Word can occur in many $\mathrm{Collocations}$. Resampling a single $\mathrm{Collocation}$ can change the way it is analysed into Words, thus changing the analysis of all of the $\mathrm{Sentences}$ containing that $\mathrm{Collocation}$.

Table label resampling is an additional resampling step performed after each Gibbs sweep through the training data in which we resample the parse trees labeling the tables in the adaptor for each $X \in A$. Specifically, if the adaptor C_X for $X \in A$ currently contains m tables labeled with the trees $\boldsymbol{t} = (t_1, \ldots, t_m)$ then table label resampling replaces each $t_j, j \in 1, \ldots, m$ in turn with a tree sampled from $\mathrm{P}(t \mid \boldsymbol{t}_{-j}, w_j)$, where w_j is the terminal yield of t_j. (Within each adaptor we actually resample all of the trees $\boldsymbol{t}$ in a randomly chosen order).

Table label resampling is a kind of Gibbs sweep, but at a higher level in the Bayesian hierarchy than the standard Gibbs sweep. It's easy to show that table label resampling preserves detailed balance for the adaptor grammars presented in this paper, so interposing table label resampling steps with the standard Gibbs steps also preserves detailed balance.

We expect table label resampling to have the greatest impact on models with a rich hierarchical structure, and the experimental results in Table 1 confirm this. The unigram adaptor grammar does not involve nested adapted nonterminals, so we would not expect table label resampling to have any effect on its analyses. On the other hand, the collocation-syllable adaptor grammar involves a rich hierarchical structure, and in fact without table label resampling our sampler did not burn in or mix within 2,000 iterations. As Figure 1 shows, table label resampling produces parses with higher posterior probability, and Table 1 shows that table label resampling makes a significant difference in the word segmentation f-score of the collocation and collocation-syllable adaptor grammars.

5 Conclusion

This paper has examined adaptor grammar inference procedures and their effect on the word segmentation problem. Some of the techniques investigated here, such as batch versus incremental initialization, are quite general and may be applicable to a wide range of other algorithms, but some of the other techniques, such as table label resampling, are specialized to nonparametric hierarchical Bayesian inference. We've shown that sampling adaptor hyperparameters is feasible, and demonstrated that this improves word segmentation accuracy of the collocation-syllable adaptor grammar by almost 10%, corresponding to an error reduction of over 35% compared to the best results presented in Johnson (2008). We also described and investigated table label resampling, which dramatically improves the effectiveness of Gibbs sampling estimators for complex adaptor grammars, and makes it possible to work with adaptor grammars with complex hierarchical structure.

Acknowledgments

We thank Erik Sudderth for suggesting sampling the Pitman-Yor hyperparameters and the ACL reviewers for their insightful comments. This research was funded by NSF awards 0544127 and 0631667 to Mark Johnson.

References

N. Bernstein-Ratner. 1987. The phonology of parent-child speech. In K. Nelson and A. van Kleeck, editors, *Children's Language*, volume 6. Erlbaum, Hillsdale, NJ.

Daniel Blanchard and Jeffrey Heinz. 2008. Improving word segmentation by simultaneously learning phonotactics. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 65–72, Manchester, England, August.

David Blei, Thomas L. Griffiths, Michael I. Jordan, and Joshua B. Tenenbaum. 2004. Hierarchical topic models and the nested chinese restaurant process. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA.

Rens Bod. 1998. *Beyond grammar: an experience-based theory of language*. CSLI Publications, Stanford, California.

M. Brent and T. Cartwright. 1996. Distributional regularity and phonotactic constraints are useful for segmentation. *Cognition*, 61:93–125.

M. Brent. 1999. An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*, 34:71–105.

Jeffrey Elman. 1990. Finding structure in time. *Cognitive Science*, 14:197–211.

Margaret M. Fleck. 2008. Lexicalized phonotactic word segmentation. In *Proceedings of ACL-08: HLT*, pages 130–138, Columbus, Ohio, June. Association for Computational Linguistics.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2006a. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 673–680, Sydney, Australia. Association for Computational Linguistics.

Sharon Goldwater, Tom Griffiths, and Mark Johnson. 2006b. Interpolating between types and tokens by estimating power-law generators. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 459–466, Cambridge, MA. MIT Press.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2007. Distributional cues to word boundaries: Context is important. In David Bamman, Tatiana Magnitskaia, and Colleen Zaller, editors, *Proceedings of the 31st Annual Boston University Conference on Language Development*, pages 239–250, Somerville, MA. Cascadilla Press.

H. Ishwaran and L. F. James. 2003. Generalized weighted Chinese restaurant processes for species sampling mixture models. *Statistica Sinica*, 13:1211–1235.

Mark Johnson, Thomas Griffiths, and Sharon Goldwater. 2007a. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 139–146, Rochester, New York, April. Association for Computational Linguistics.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007b. Adaptor Grammars: A framework for specifying compositional nonparametric Bayesian models. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 641–648. MIT Press, Cambridge, MA.

Mark Johnson. 2008. Using adaptor grammars to identifying synergies in the unsupervised acquisition of linguistic structure. In *Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics*, Columbus, Ohio. Association for Computational Linguistics.

Aravind Joshi. 2003. Tree adjoining grammars. In Ruslan Mikkov, editor, *The Oxford Handbook of Computational Linguistics*, pages 483–501. Oxford University Press, Oxford, England.

David J.C. MacKay. 2003. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press.

Radford M. Neal. 2003. Slice sampling. *Annals of Statistics*, 31:705–767.

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

J. Pitman. 1995. Exchangeable and partially exchangeable random partitions. *Probability Theory and Related Fields*, 102:145–158.

Christian P. Robert and George Casella. 2004. *Monte Carlo Statistical Methods*. Springer.

Andreas Stolcke and Stephen Omohundro. 1994. Inducing probabilistic grammars by Bayesian model merging. In Rafael C. Carrasco and Jose Oncina, editors, *Grammatical Inference and Applications*, pages 106–118. Springer, New York.

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