Bayesian Inference for PCFGs via Markov chain Monte Carlo

Mark Johnson
Cognitive and Linguistic Sciences
Brown University
Mark_Johnson@brown.edu

Thomas L. Griffiths
Department of Psychology
University of California, Berkeley
Tom_Griffiths@berkeley.edu

Sharon Goldwater
Department of Linguistics
Stanford University
sgwater@stanford.edu

Abstract

This paper presents two Markov chain Monte Carlo (MCMC) algorithms for Bayesian inference of probabilistic context free grammars (PCFGs) from terminal strings, providing an alternative to maximum-likelihood estimation using the Inside-Outside algorithm. We illustrate these methods by estimating a sparse grammar describing the morphology of the Bantu language Sesotho, demonstrating that with suitable priors Bayesian techniques can infer linguistic structure in situations where maximum likelihood methods such as the Inside-Outside algorithm only produce a trivial grammar.

1 Introduction

The standard methods for inferring the parameters of probabilistic models in computational linguistics are based on the principle of maximum-likelihood estimation; for example, the parameters of Probabilistic Context-Free Grammars (PCFGs) are typically estimated from strings of terminals using the Inside-Outside (IO) algorithm, an instance of the Expectation Maximization (EM) procedure (Lari and Young, 1990). However, much recent work in machine learning and statistics has turned away from maximum-likelihood in favor of Bayesian methods, and there is increasing interest in Bayesian methods in computational linguistics as well (Finkel et al., 2006). This paper presents two Markov chain Monte Carlo (MCMC) algorithms for inferring PCFGs and their parses from strings alone. These can be viewed as Bayesian alternatives to the IO algorithm.

The goal of Bayesian inference is to compute a distribution over plausible parameter values. This "posterior" distribution is obtained by combining the likelihood with a "prior" distribution $\mathrm{P}(\theta)$ over parameter values θ. In the case of PCFG inference θ is the vector of rule probabilities, and the prior might assert a preference for a sparse grammar (see below). The posterior probability of each value of θ is given by Bayes' rule:

$$\mathrm{P}(\theta|D) \quad \propto \quad \mathrm{P}(D|\theta)\mathrm{P}(\theta). \tag{1}$$

In principle Equation 1 defines the posterior probability of any value of θ, but computing this may not be tractable analytically or numerically. For this reason a variety of methods have been developed to support approximate Bayesian inference. One of the most popular methods is Markov chain Monte Carlo (MCMC), in which a Markov chain is used to sample from the posterior distribution.

This paper presents two new MCMC algorithms for inferring the posterior distribution over parses and rule probabilities given a corpus of strings. The first algorithm is a component-wise Gibbs sampler which is very similar in spirit to the EM algorithm, drawing parse trees conditioned on the current parameter values and then sampling the parameters conditioned on the current set of parse trees. The second algorithm is a component-wise Hastings sampler that "collapses" the probabilistic model, integrating over the rule probabilities of the PCFG, with the goal of speeding convergence. Both algorithms use an efficient dynamic programming technique to sample parse trees.

Given their usefulness in other disciplines, we believe that Bayesian methods like these are likely to be of general utility in computational linguistics as well. As a simple illustrative example, we use these methods to infer morphological parses for verbs from Sesotho, a southern Bantu language with agglutinating morphology. Our results illustrate that Bayesian inference using a prior that favors sparsity can produce linguistically reasonable analyses in situations in which EM does not.

The rest of this paper is structured as follows. The next section introduces the background for our paper, summarizing the key ideas behind PCFGs, Bayesian inference, and MCMC. Section 3 introduces our first MCMC algorithm, a Gibbs sampler for PCFGs. Section 4 describes an algorithm for sampling trees from the distribution over trees defined by a PCFG. Section 5 shows how to integrate out the rule weight parameters θ in a PCFG, allowing us to sample directly from the posterior distribution over parses for a corpus of strings. Finally, Section 6 illustrates these methods in learning Sesotho morphology.

2 Background

2.1 Probabilistic context-free grammars

Let $G = (T, N, S, R)$ be a Context-Free Grammar in Chomsky normal form with no useless productions, where T is a finite set of *terminal symbols*, N is a finite set of *nonterminal symbols* (disjoint from T), $S \in N$ is a distinguished nonterminal called the *start symbol*, and R is a finite set of *productions* of the form $A \rightarrow B\,C$ or $A \rightarrow w$, where $A, B, C \in N$ and $w \in T$. In what follows we use β as a variable ranging over $(N \times N) \cup T$.

A *Probabilistic Context-Free Grammar* (G, θ) is a pair consisting of a context-free grammar G and a real-valued vector θ of length $|R|$ indexed by productions, where $\theta_{A\rightarrow\beta}$ is the *production probability* associated with the production $A \rightarrow \beta \in R$. We require that $\theta_{A\rightarrow\beta} \geq 0$ and that for all nonterminals $A \in N$, $\sum_{A\rightarrow\beta\in R} \theta_{A\rightarrow\beta} = 1$.

A PCFG (G, θ) defines a probability distribution over trees t as follows:

$$\mathrm{P}_G(t|\theta) \quad = \quad \prod_{r \in R} \theta_r^{f_r(t)}$$

where t is generated by G and $f_r(t)$ is the number of times the production $r = A \rightarrow \beta \in R$ is used in the derivation of t. If G does not generate t let $\mathrm{P}_G(t|\theta) = 0$. The *yield* $y(t)$ of a parse tree t is the sequence of terminals labeling its leaves. The probability of a string $w \in T^+$ of terminals is the sum of the probability of all trees with yield w, i.e.:

$$\mathrm{P}_G(w|\theta) \quad = \quad \sum_{t:y(t)=w} \mathrm{P}_G(t|\theta).$$

2.2 Bayesian inference for PCFGs

Given a corpus of strings $\mathbf{w} = (w_1, \ldots, w_n)$, where each w_i is a string of terminals generated by a known CFG G, we would like to be able to infer the production probabilities θ that best describe that corpus. Taking $\mathbf{w}$ to be our data, we can apply Bayes' rule (Equation 1) to obtain:

$$\begin{aligned} \mathrm{P}(\theta|\mathbf{w}) &\propto \mathrm{P}_G(\mathbf{w}|\theta)\mathrm{P}(\theta), \quad \text{where} \\ \mathrm{P}_G(\mathbf{w}|\theta) &= \prod_{i=1}^{n} \mathrm{P}_G(w_i|\theta). \end{aligned}$$

Using $\mathbf{t}$ to denote a sequence of parse trees for $\mathbf{w}$, we can compute the joint posterior distribution over $\mathbf{t}$ and θ, and then marginalize over $\mathbf{t}$, with $\mathrm{P}(\theta|\mathbf{w}) = \sum_{\mathbf{t}} \mathrm{P}(\mathbf{t}, \theta|\mathbf{w})$. The joint posterior distribution on $\mathbf{t}$ and θ is given by:

$$\begin{aligned} \mathrm{P}(\mathbf{t}, \theta|\mathbf{w}) &\propto \mathrm{P}(\mathbf{w}|\mathbf{t})\mathrm{P}(\mathbf{t}|\theta)\mathrm{P}(\theta) \\ &= \left(\prod_{i=1}^{n} \mathrm{P}(w_i|t_i)\mathrm{P}(t_i|\theta)\right) \mathrm{P}(\theta) \end{aligned}$$

with $\mathrm{P}(w_i|t_i) = 1$ if $y(t_i) = w_i$, and 0 otherwise.

2.3 Dirichlet priors

The first step towards computing the posterior distribution is to define a prior on θ. We take $\mathrm{P}(\theta)$ to be a product of Dirichlet distributions, with one distribution for each non-terminal $A \in N$. The prior is parameterized by a positive real valued vector α indexed by productions R, so each production probability $\theta_{A\rightarrow\beta}$ has a corresponding Dirichlet parameter $\alpha_{A\rightarrow\beta}$. Let R_A be the set of productions in R

with left-hand side A, and let θ_A and α_A refer to the component subvectors of θ and α respectively indexed by productions in R_A. The Dirichlet prior $\mathrm{P}_D(\theta|\alpha)$ is:

$$\begin{aligned}
\mathrm{P}_D(\theta|\alpha) &= \prod_{A\in N} \mathrm{P}_D(\theta_A|\alpha_A), \quad \text{where} \\
\mathrm{P}_D(\theta_A|\alpha_A) &= \frac{1}{C(\alpha_A)} \prod_{r\in R_A} \theta_r^{\alpha_r - 1} \quad \text{and} \\
C(\alpha_A) &= \frac{\prod_{r\in R_A}\Gamma(\alpha_r)}{\Gamma(\sum_{r\in R_A}\alpha_r)} \qquad (2)
\end{aligned}$$

where Γ is the generalized factorial function and $C(\alpha)$ is a normalization constant that does not depend on θ_A.

Dirichlet priors are useful because they are *conjugate* to the distribution over trees defined by a PCFG. This means that the posterior distribution on θ given a set of parse trees, $\mathrm{P}(\theta|\mathbf{t},\alpha)$, is also a Dirichlet distribution. Applying Bayes' rule,

$$\begin{aligned}
\mathrm{P}_G(\theta|\mathbf{t},\alpha) &\propto \mathrm{P}_G(\mathbf{t}|\theta)\,\mathrm{P}_D(\theta|\alpha) \\
&\propto \left(\prod_{r\in R}\theta_r^{f_r(\mathbf{t})}\right)\left(\prod_{r\in R}\theta_r^{\alpha_r-1}\right) \\
&= \prod_{r\in R}\theta_r^{f_r(\mathbf{t})+\alpha_r-1}
\end{aligned}$$

which is a Dirichlet distribution with parameters $\mathbf{f}(\mathbf{t})+\alpha$, where $\mathbf{f}(\mathbf{t})$ is the vector of production counts in $\mathbf{t}$ indexed by $r \in R$. We can thus write:

$$\mathrm{P}_G(\theta|\mathbf{t},\alpha) = \mathrm{P}_D(\theta|\mathbf{f}(\mathbf{t})+\alpha)$$

which makes it clear that the production counts combine directly with the parameters of the prior.

2.4 Markov chain Monte Carlo

Having defined a prior on θ, the posterior distribution over $\mathbf{t}$ and θ is fully determined by a corpus $\mathbf{w}$. Unfortunately, computing the posterior probability of even a single choice of $\mathbf{t}$ and θ is intractable, as evaluating the normalizing constant for this distribution requires summing over all possible parses for the entire corpus and all sets of production probabilities. Nonetheless, it is possible to define algorithms that sample from this distribution using Markov chain Monte Carlo (MCMC).

MCMC algorithms construct a Markov chain whose states $s \in \mathcal{S}$ are the objects we wish to sample. The state space $\mathcal{S}$ is typically astronomically large — in our case, the state space includes all possible parses of the entire training corpus $\mathbf{w}$ — and the transition probabilities $\mathrm{P}(s'|s)$ are specified via a scheme guaranteed to converge to the desired distribution $\pi(s)$ (in our case, the posterior distribution). We "run" the Markov chain (i.e., starting in initial state s_0, sample a state s_1 from $\mathrm{P}(s'|s_0)$, then sample state s_2 from $\mathrm{P}(s'|s_1)$, and so on), with the probability that the Markov chain is in a particular state, $\mathrm{P}(s_i)$, converging to $\pi(s_i)$ as $i \to \infty$.

After the chain has run long enough for it to approach its stationary distribution, the expectation $\mathrm{E}_\pi[f]$ of any function $f(s)$ of the state s will be approximated by the average of that function over the set of sample states produced by the algorithm. For example, in our case, given samples (t_i, θ_i) for $i = 1, \ldots, \ell$ produced by an MCMC algorithm, we can estimate θ as

$$\mathrm{E}_\pi[\theta] \approx \frac{1}{\ell}\sum_{i=1}^{\ell}\theta_i$$

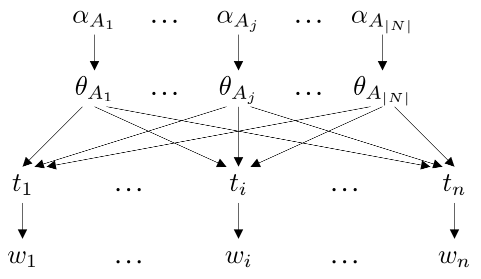
The remainder of this paper presents two MCMC algorithms for PCFGs. Both algorithms proceed by setting the initial state of the Markov chain to a guess for $(\mathbf{t},\theta)$ and then sampling successive states using a particular transition matrix. The key difference betwen the two algorithms is the form of the transition matrix they assume.

3 A Gibbs sampler for $\mathrm{P}(\mathbf{t},\theta|\mathbf{w},\alpha)$

The Gibbs sampler (Geman and Geman, 1984) is one of the simplest MCMC methods, in which transitions between states of the Markov chain result from sampling each component of the state conditioned on the current value of all other variables. In our case, this means alternating between sampling from two distributions:

$$\begin{aligned}
\mathrm{P}(\mathbf{t}|\theta,\mathbf{w},\alpha) &= \prod_{i=1}^{n}\mathrm{P}(t_i|w_i,\theta), \quad \text{and} \\
\mathrm{P}(\theta|\mathbf{t},\mathbf{w},\alpha) &= \mathrm{P}_D(\theta|\mathbf{f}(\mathbf{t})+\alpha) \\
&= \prod_{A\in N}\mathrm{P}_D(\theta_A|\mathbf{f}_A(\mathbf{t})+\alpha_A).
\end{aligned}$$

Thus every two steps we generate a new sample of $\mathbf{t}$ and θ. This alternation between parsing and updating θ is reminiscent of the EM algorithm, with



α_{A_1} … α_{A_j} … $\alpha_{A_{|N|}}$
θ_{A_1} … θ_{A_j} … $\theta_{A_{|N|}}$
t_1 … t_i … t_n
w_1 … w_i … w_n

Figure 1: A Bayes net representation of dependencies among the variables in a PCFG.

the Expectation step replaced by sampling $\mathbf{t}$ and the Maximization step replaced by sampling θ.

The dependencies among variables in a PCFG are depicted graphically in Figure 1, which makes clear that the Gibbs sampler is highly parallelizable (just like the EM algorithm). Specifically, the parses t_i are independent given θ and so can be sampled in parallel from the following distribution as described in the next section.

$$\mathrm{P}_G(t_i|w_i,\theta) \quad = \quad \frac{\mathrm{P}_G(t_i|\theta)}{\mathrm{P}_G(w_i|\theta)}$$

We make use of the fact that the posterior is a product of independent Dirichlet distributions in order to sample θ from $\mathrm{P}_D(\theta|\mathbf{t},\alpha)$. The production probabilities θ_A for each nonterminal $A \in N$ are sampled from a Dirchlet distibution with parameters $\alpha'_A = f_A(\mathbf{t}) + \alpha_A$. There are several methods for sampling $\theta = (\theta_1, \ldots, \theta_m)$ from a Dirichlet distribution with parameters $\alpha = (\alpha_1, \ldots, \alpha_m)$, with the simplest being sampling x_j from a Gamma(α_j) distribution for $j = 1, \ldots, m$ and then setting $\theta_j = x_j / \sum_{k=1}^m x_k$ (Gentle, 2003).

4 Efficiently sampling from $\mathrm{P}(t|w,\theta)$

This section completes the description of the Gibbs sampler for $(\mathbf{t},\theta)$ by describing a dynamic programming algorithm for sampling trees from the set of parses for a string generated by a PCFG. This algorithm appears fairly widely known: it was described by Goodman (1998) and Finkel et al (2006) and used by Ding et al (2005), and is very similar to other dynamic programming algorithms for CFGs, so we only summarize it here. The algorithm consists of two steps. The first step constructs a standard "inside" table or chart, as used in the Inside-Outside algorithm for PCFGs (Lari and Young, 1990). The second step involves a recursion from larger to smaller strings, sampling from the productions that expand each string and constructing the corresponding tree in a top-down fashion.

In this section we take w to be a string of terminal symbols $w = (w_1, \ldots, w_n)$ where each $w_i \in T$, and define $w_{i,k} = (w_{i+1}, \ldots, w_k)$ (i.e., the substring from w_{i+1} up to w_k). Further, let $G_A = (T, N, A, R)$, i.e., a CFG just like G except that the start symbol has been replaced with A, so, $\mathrm{P}_{G_A}(t|\theta)$ is the probability of a tree t whose root node is labeled A and $\mathrm{P}_{G_A}(w|\theta)$ is the sum of the probabilities of all trees whose root nodes are labeled A with yield w.

The Inside algorithm takes as input a PCFG (G,θ) and a string $w = w_{0,n}$ and constructs a table with entries $p_{A,i,k}$ for each $A \in N$ and $0 \leq i < k \leq n$, where $p_{A,i,k} = \mathrm{P}_{G_A}(w_{i,k}|\theta)$, i.e., the probability of A rewriting to $w_{i,k}$. The table entries are recursively defined below, and computed by enumerating all feasible i, k and A in any order such that all smaller values of $k-i$ are enumerated before any larger values.

$$\begin{aligned} p_{A,k-1,k} &= \theta_{A \to w_k} \\ p_{A,i,k} &= \sum_{A \to B\,C \in R} \; \sum_{i<j<k} \theta_{A \to B\,C}\, p_{B,i,j}\, p_{C,j,k} \end{aligned}$$

for all $A, B, C \in N$ and $0 \leq i < j < k \leq n$. At the end of the Inside algorithm, $\mathrm{P}_G(w|\theta) = p_{S,0,n}$.

The second step of the sampling algorithm uses the function SAMPLE, which returns a sample from $\mathrm{P}_G(t|w,\theta)$ given the PCFG (G,θ) and the inside table $p_{A,i,k}$. SAMPLE takes as arguments a nonterminal $A \in N$ and a pair of string positions $0 \leq i < k \leq n$ and returns a tree drawn from $\mathrm{P}_{G_A}(t|w_{i,k},\theta)$. It functions in a top-down fashion, selecting the production $A \to B\,C$ to expand the A, and then recursively calling itself to expand B and C respectively.

```
function SAMPLE$(A, i, k)$ :
if $k - i = 1$ then return TREE$(A, w_k)$
$(j, B, C)$ = MULTI$(A, i, k)$
return TREE$(A$, SAMPLE$(B, i, j)$, SAMPLE$(C, j, k))$
```

In this pseudo-code, TREE is a function that constructs unary or binary tree nodes respectively, and

MULTI is a function that produces samples from a multinomial distribution over the possible "split" positions j and nonterminal children B and C, where:

$$\mathrm{P}(j, B, C) = \frac{\theta_{A \to B\,C}\, \mathrm{P}_{G_B}(w_{i,j}|\theta)\, \mathrm{P}_{G_C}(w_{j,k}|\theta)}{\mathrm{P}_{G_A}(w_{i,k}|\theta)}$$

5 A Hastings sampler for $\mathrm{P}(\mathbf{t}|\mathbf{w}, \alpha)$

The Gibbs sampler described in Section 3 has the disadvantage that each sample of θ requires reparsing the training corpus $\mathbf{w}$. In this section, we describe a component-wise Hastings algorithm for sampling directly from $\mathrm{P}(\mathbf{t}|\mathbf{w}, \alpha)$, marginalizing over the production probabilities θ. Transitions between states are produced by sampling parses t_i from $\mathrm{P}(t_i|w_i, \mathbf{t}_{-i}, \alpha)$ for each string w_i in turn, where $\mathbf{t}_{-i} = (t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n)$ is the current set of parses for $\mathbf{w}_{-i} = (w_1, \ldots, w_{i-1}, w_{i+1}, \ldots, w_n)$. Marginalizing over θ effectively means that the production probabilities are updated after each sentence is parsed, so it is reasonable to expect that this algorithm will converge faster than the Gibbs sampler described earlier. While the sampler does not explicitly provide samples of θ, the results outlined in Sections 2.3 and 3 can be used to sample the posterior distribution over θ for each sample of $\mathbf{t}$ if required.

Let $\mathrm{P}_D(\theta|\alpha)$ be a Dirichlet product prior, and let Δ be the probability simplex for θ. Then by integrating over the posterior Dirichlet distributions we have:

$$\begin{aligned} \mathrm{P}(\mathbf{t}|\alpha) &= \int_\Delta \mathrm{P}_G(\mathbf{t}|\theta)\mathrm{P}_D(\theta|\alpha)d\theta \\ &= \prod_{A \in N} \frac{C(\alpha_A + \mathbf{f}_A(\mathbf{t}))}{C(\alpha_A)} \end{aligned} \quad (3)$$

where C was defined in Equation 2. Because we are marginalizing over θ, the trees t_i become dependent upon one another. Intuitively, this is because w_i may provide information about θ that influences how some other string w_j should be parsed.

We can use Equation 3 to compute the conditional probability $\mathrm{P}(t_i|\mathbf{t}_{-i}, \alpha)$ as follows:

$$\begin{aligned} \mathrm{P}(t_i|\mathbf{t}_{-i}, \alpha) &= \frac{\mathrm{P}(\mathbf{t}|\alpha)}{\mathrm{P}(\mathbf{t}_{-i}|\alpha)} \\ &= \prod_{A \in N} \frac{C(\alpha_A + \mathbf{f}_A(\mathbf{t}))}{C(\alpha_A + \mathbf{f}_A(\mathbf{t}_{-i}))} \end{aligned}$$

Now, if we could sample from

$$\mathrm{P}(t_i|w_i, \mathbf{t}_{-i}, \alpha) = \frac{\mathrm{P}(w_i|t_i)\mathrm{P}(t_i|\mathbf{t}_{-i}, \alpha)}{\mathrm{P}(w_i|\mathbf{t}_{-i}, \alpha)}$$

we could construct a Gibbs sampler whose states were the parse trees $\mathbf{t}$. Unfortunately, we don't even know if there is an efficient algorithm for calculating $\mathrm{P}(w_i|\mathbf{t}_{-i}, \alpha)$, let alone an efficient sampling algorithm for this distribution.

Fortunately, this difficulty is not fatal. A Hastings sampler for a probability distribution $\pi(s)$ is an MCMC algorithm that makes use of a *proposal distribution* $Q(s'|s)$ from which it draws samples, and uses an acceptance/rejection scheme to define a transition kernel with the desired distribution $\pi(s)$. Specifically, given the current state s, a sample $s' \neq s$ drawn from $Q(s'|s)$ is accepted as the next state with probability

$$A(s, s') = \min\left\{1, \frac{\pi(s')Q(s|s')}{\pi(s)Q(s'|s)}\right\}$$

and with probability $1 - A(s, s')$ the proposal is rejected and the next state is the current state s.

We use a component-wise proposal distribution, generating new proposed values for t_i, where i is chosen at random. Our proposal distribution is the posterior distribution over parse trees generated by the PCFG with grammar G and production probabilities θ', where θ' is chosen based on the current $\mathbf{t}_{-i}$ as described below. Each step of our Hastings sampler is as follows. First, we compute θ' from $\mathbf{t}_{-i}$ as described below. Then we sample t'_i from $\mathrm{P}(t_i|w_i, \theta')$ using the algorithm described in Section 4. Finally, we accept the proposal t'_i given the old parse t_i for w_i with probability:

$$\begin{aligned} A(t_i, t'_i) &= \min\left\{1, \frac{\mathrm{P}(t'_i|w_i, \mathbf{t}_{-i}, \alpha)\mathrm{P}(t_i|w_i, \theta')}{\mathrm{P}(t_i|w_i, \mathbf{t}_{-i}, \alpha)\mathrm{P}(t'_i|w_i, \theta')}\right\} \\ &= \min\left\{1, \frac{\mathrm{P}(t'_i|\mathbf{t}_{-i}, \alpha)\mathrm{P}(t_i|w_i, \theta')}{\mathrm{P}(t_i|\mathbf{t}_{-i}, \alpha)\mathrm{P}(t'_i|w_i, \theta')}\right\} \end{aligned}$$

The key advantage of the Hastings sampler over the Gibbs sampler here is that because the acceptance probability is a ratio of probabilities, the difficult to compute $\mathrm{P}(w_i|\mathbf{t}_{-i},\alpha)$ is a common factor of both the numerator and denominator, and hence is not required. The $P(w_i|t_i)$ term also disappears, being 1 for both the numerator and the denominator since our proposal distribution can only generate trees for which w_i is the yield.

All that remains is to specify the production probabilities θ' of the proposal distribution $\mathrm{P}(t'_i|w_i,\theta')$. While the acceptance rule used in the Hastings algorithm ensures that it produces samples from $\mathrm{P}(t_i|w_i,\mathbf{t}_{-i},\alpha)$ with any proposal grammar θ' in which all productions have nonzero probability, the algorithm is more efficient (i.e., fewer proposals are rejected) if the proposal distribution is close to the distribution to be sampled.

Given the observations above about the correspondence between terms in $\mathrm{P}(t_i|\mathbf{t}_{-i},\alpha)$ and the relative frequency of the corresponding productions in $\mathbf{t}_{-i}$, we set θ' to the expected value $\mathrm{E}[\theta|\mathbf{t}_{-i},\alpha]$ of θ given $\mathbf{t}_{-i}$ and α as follows:

$$\theta'_r = \frac{f_r(\mathbf{t}_{-i}) + \alpha_r}{\sum_{r' \in R_A} f_{r'}(\mathbf{t}_{-i}) + \alpha_{r'}}$$

6 Inferring sparse grammars

As stated in the introduction, the primary contribution of this paper is introducing MCMC methods for Bayesian inference to computational linguistics. Bayesian inference using MCMC is a technique of generic utility, much like Expectation-Maximization and other general inference techniques, and we believe that it belongs in every computational linguist's toolbox alongside these other techniques.

Inferring a PCFG to describe the syntactic structure of a natural language is an obvious application of grammar inference techniques, and it is well-known that PCFG inference using maximum-likelihood techniques such as the Inside-Outside (IO) algorithm, a dynamic programming Expectation-Maximization (EM) algorithm for PCFGs, performs extremely poorly on such tasks. We have applied the Bayesian MCMC methods described here to such problems and obtain results very similar to those produced using IO. We believe that the primary reason why both IO and the Bayesian methods perform so poorly on this task is that simple PCFGs are not accurate models of English syntactic structure. We know that PCFGs

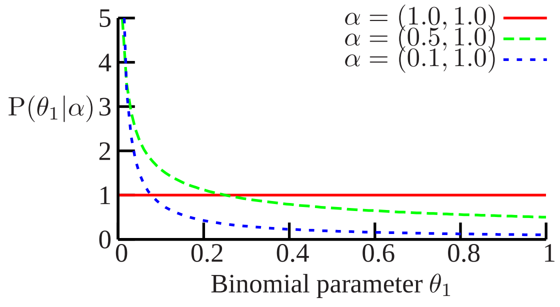


Figure 2: A Dirichlet prior α on a binomial parameter θ_1. As $\alpha_1 \to 0$, $\mathrm{P}(\theta_1|\alpha)$ is increasingly concentrated around 0.

that represent only major phrasal categories ignore a wide variety of lexical and syntactic dependencies in natural language. State-of-the-art systems for unsupervised syntactic structure induction system uses models that are very different to these kinds of PCFGs (Klein and Manning, 2004; Smith and Eisner, 2006).[1]

Our goal in this section is modest: we aim merely to provide an illustrative example of Bayesian inference using MCMC. As Figure 2 shows, when the Dirichlet prior parameter α_r approaches 0 the prior probability $\mathrm{P}_D(\theta_r|\alpha)$ becomes increasingly concentrated around 0. This ability to bias the sampler toward sparse grammars (i.e., grammars in which many productions have probabilities close to 0) is useful when we attempt to identify relevant productions from a much larger set of possible productions via parameter estimation.

The Bantu language Sesotho is a richly agglutinative language, in which verbs consist of a sequence of morphemes, including optional Subject Markers (SM), Tense (T), Object Markers (OM), Mood (M) and derivational affixes as well as the obligatory Verb stem (V), as shown in the following example:

re	*-a*	*-di*	*-bon*	*-a*
SM	T	OM	V	M

"We see them"

[1]It is easy to demonstrate that the poor quality of the PCFG models is the cause of these problems rather than search or other algorithmic issues. If one initializes either the IO or Bayesian estimation procedures with treebank parses and then runs the procedure using the yields alone, the accuracy of the parses uniformly decreases while the (posterior) likelihood uniformly increases with each iteration, demonstrating that improving the (posterior) likelihood of such models does not improve parse accuracy.

We used an implementation of the Hastings sampler described in Section 5 to infer morphological parses $\mathbf{t}$ for a corpus $\mathbf{w}$ of 2,283 unsegmented Sesotho verb types extracted from the Sesotho corpus available from CHILDES (MacWhinney and Snow, 1985; Demuth, 1992). We chose this corpus because the words have been morphologically segmented manually, making it possible for us to evaluate the morphological parses produced by our system. We constructed a CFG G containing the following productions

$$\begin{array}{lcl} \mathsf{Word} & \rightarrow & \mathsf{V} \\ \mathsf{Word} & \rightarrow & \mathsf{V\ M} \\ \mathsf{Word} & \rightarrow & \mathsf{SM\ V\ M} \\ \mathsf{Word} & \rightarrow & \mathsf{SM\ T\ V\ M} \\ \mathsf{Word} & \rightarrow & \mathsf{SM\ T\ OM\ V\ M} \end{array}$$

together with productions expanding the preterminals $\mathsf{SM}, \mathsf{T}, \mathsf{OM}, \mathsf{V}$ and M to each of the 16,350 distinct substrings occuring anywhere in the corpus, producting a grammar with 81,755 productions in all. In effect, G encodes the basic morphological structure of the Sesotho verb (ignoring factors such as derivation morphology and irregular forms), but provides no information about the phonological identity of the morphemes.

Note that G actually generates a *finite* language. However, G parameterizes the probability distribution over the strings it generates in a manner that would be difficult to succintly characterize except in terms of the productions given above. Moreover, with approximately 20 times more productions than training strings, each string is highly ambiguous and estimation is highly underconstrained, so it provides an excellent test-bed for sparse priors.

We estimated the morphological parses $\mathbf{t}$ in two ways. First, we ran the IO algorithm initialized with a uniform initial estimate θ_0 for θ to produce an estimate of the MLE $\hat{\theta}$, and then computed the Viterbi parses $\hat{\mathbf{t}}$ of the training corpus $\mathbf{w}$ with respect to the PCFG $(G, \hat{\theta})$. Second, we ran the Hastings sampler initialized with trees sampled from (G, θ_0) with several different values for the parameters of the prior. We experimented with a number of techniques for speeding convergence of both the IO and Hastings algorithms, and two of these were particularly effective on this problem. Annealing, i.e., using $\mathrm{P}(\mathbf{t}|\mathbf{w})^{1/\tau}$ in place of $\mathrm{P}(\mathbf{t}|\mathbf{w})$ where τ is a "temperature" parameter starting around 5 and slowly adjusted toward 1, sped the convergence of both algorithms. We ran both algorithms for several thousand iterations over the corpus, and both seemed to converge fairly quickly once τ was set to 1. "Jittering" the initial estimate of θ used in the IO algorithm also sped its convergence.

The IO algorithm converges to a solution where $\theta_{\mathsf{Word} \rightarrow \mathsf{V}} = 1$, and every string $w \in \mathbf{w}$ is analysed as a single morpheme V. (In fact, in this grammar $\mathrm{P}(w_i|\theta)$ is the empirical probability of w_i, and it is easy to prove that this θ is the MLE).

The samples $\mathbf{t}$ produced by the Hastings algorithm depend on the parameters of the Dirichlet prior. We set α_r to a single value α for all productions r. We found that for $\alpha > 10^{-2}$ the samples produced by the Hastings algorithm were the same trivial analyses as those produced by the IO algorithm, but as α was reduced below this $\mathbf{t}$ began to exhibit nontrivial structure. We evaluated the quality of the segmentations in the morphological analyses $\mathbf{t}$ in terms of unlabeled precision, recall, f-score and exact match (the fraction of words correctly segmented into morphemes; we ignored morpheme labels because the manual morphological analyses contain many morpheme labels that we did not include in G). Figure 3 contains a plot of how these quantities vary with α; obtaining an f-score of 0.75 and an exact word match accuracy of 0.54 at $\alpha = 10^{-5}$ (the corresponding values for the MLE $\hat{\theta}$ are both 0). Note that we obtained good results as α was varied over several orders of magnitude, so the actual value of α is not critical. Thus in this application the ability to prefer sparse grammars enables us to find linguistically meaningful analyses. This ability to find linguistically meaningful structure is relatively rare in our experience with unsupervised PCFG induction.

We also experimented with a version of IO modified to perform Bayesian MAP estimation, where the Maximization step of the IO procedure is replaced with Bayesian inference using a Dirichlet prior, i.e., where the rule probabilities $\theta^{(k)}$ at iteration k are estimated using:

$$\theta_r^{(k)} \quad \propto \quad \max(0, \mathrm{E}[f_r|\mathbf{w}, \theta^{(k-1)}] + \alpha - 1).$$

Clearly such an approach is very closely related to the Bayesian procedures presented in this article,

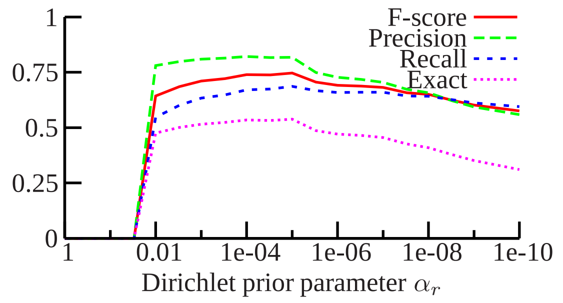


Figure 3: Accuracy of morphological segmentations of Sesotho verbs proposed by the Hastings algorithms as a function of Dirichlet prior parameter α. F-score, precision and recall are unlabeled morpheme scores, while Exact is the fraction of words correctly segmented.

and in some circumstances this may be a useful estimator. However, in our experiments with the Sesotho data above we found that for the small values of α necessary to obtain a sparse solution,the expected rule count $\mathrm{E}[f_r]$ for many rules r was less than $1-\alpha$. Thus on the next iteration $\theta_r = 0$, resulting in there being no parse whatsoever for many of the strings in the training data. Variational Bayesian techniques offer a systematic way of dealing with these problems, but we leave this for further work.

7 Conclusion

This paper has described basic algorithms for performing Bayesian inference over PCFGs given terminal strings. We presented two Markov chain Monte Carlo algorithms (a Gibbs and a Hastings sampling algorithm) for sampling from the posterior distribution over parse trees given a corpus of their yields and a Dirichlet product prior over the production probabilities. As a component of these algorithms we described an efficient dynamic programming algorithm for sampling trees from a PCFG which is useful in its own right. We used these sampling algorithms to infer morphological analyses of Sesotho verbs given their strings (a task on which the standard Maximum Likelihood estimator returns a trivial and linguistically uninteresting solution), achieving 0.75 unlabeled morpheme f-score and 0.54 exact word match accuracy. Thus this is one of the few cases we are aware of in which a PCFG estimation procedure returns linguistically meaningful structure. We attribute this to the ability of the Bayesian prior to prefer sparse grammars.

We expect that these algorithms will be of interest to the computational linguistics community both because a Bayesian approach to PCFG estimation is more flexible than the Maximum Likelihood methods that currently dominate the field (c.f., the use of a prior as a bias towards sparse solutions), and because these techniques provide essential building blocks for more complex models.

References

Katherine Demuth. 1992. Acquisition of Sesotho. In Dan Slobin, editor, *The Cross-Linguistic Study of Language Acquisition*, volume 3, pages 557–638. Lawrence Erlbaum Associates, Hillsdale, N.J.

Ye Ding, Chi Yu Chan, and Charles E. Lawrence. 2005. RNA secondary structure prediction by centroids in a Boltzmann weighted ensemble. *RNA*, 11:1157–1166.

Jenny Rose Finkel, Christopher D. Manning, and Andrew Y. Ng. 2006. Solving the problem of cascading errors: Approximate Bayesian inference for linguistic annotation pipelines. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 618–626, Sydney, Australia. Association for Computational Linguistics.

Stuart Geman and Donald Geman. 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741.

James E. Gentle. 2003. *Random number generation and Monte Carlo methods*. Springer, New York, 2nd edition.

Joshua Goodman. 1998. *Parsing inside-out*. Ph.D. thesis, Harvard University. available from http://research.microsoft.com/˜joshuago/.

Dan Klein and Chris Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 478–485.

K. Lari and S.J. Young. 1990. The estimation of Stochastic Context-Free Grammars using the Inside-Outside algorithm. *Computer Speech and Language*, 4(35-56).

Brian MacWhinney and Catherine Snow. 1985. The child language data exchange system. *Journal of Child Language*, 12:271–296.

Noah A. Smith and Jason Eisner. 2006. Annealing structural bias in multilingual weighted grammar induction. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 569–576, Sydney, Australia. Association for Computational Linguistics.