A comparison of Bayesian estimators for unsupervised Hidden Markov Model POS taggers

Jianfeng Gao
Microsoft Research
Redmond, WA, USA
jfgao@microsoft.com

Mark Johnson
Brown Univeristy
Providence, RI, USA
Mark_Johnson@Brown.edu

Abstract

There is growing interest in applying Bayesian techniques to NLP problems. There are a number of different estimators for Bayesian models, and it is useful to know what kinds of tasks each does well on. This paper compares a variety of different Bayesian estimators for Hidden Markov Model POS taggers with various numbers of hidden states on data sets of different sizes. Recent papers have given contradictory results when comparing Bayesian estimators to Expectation Maximization (EM) for unsupervised HMM POS tagging, and we show that the difference in reported results is largely due to differences in the size of the training data and the number of states in the HMM. We invesigate a variety of samplers for HMMs, including some that these earlier papers did not study. We find that all of Gibbs samplers do well with small data sets and few states, and that Variational Bayes does well on large data sets and is competitive with the Gibbs samplers. In terms of times of convergence, we find that Variational Bayes was the fastest of all the estimators, especially on large data sets, and that explicit Gibbs sampler (both pointwise and sentence-blocked) were generally faster than their collapsed counterparts on large data sets.

1 Introduction

Probabilistic models now play a central role in computational linguistics. These models define a probability distribution $\mathrm{P}(\boldsymbol{x})$ over structures or analyses $\boldsymbol{x}$. For example, in the part-of-speech (POS) tagging application described in this paper, which involves predicting the part-of-speech tag t_i of each word w_i in the sentence $\boldsymbol{w} = (w_1, \ldots, w_n)$, the structure $\boldsymbol{x} = (\boldsymbol{w}, \boldsymbol{t})$ consists of the words $\boldsymbol{w}$ in a sentence together with their corresponding parts-of-speech $\boldsymbol{t} = (t_1, \ldots, t_n)$.

In general the probabilistic models used in computational linguistics have adjustable parameters $\boldsymbol{\theta}$ which determine the distribution $\mathrm{P}(\boldsymbol{x} \mid \boldsymbol{\theta})$. In this paper we focus on bitag Hidden Markov Models (HMMs). Since our goal here is to compare algorithms rather than achieve the best performance, we keep the models simple by ignoring morphology and capitalization (two very strong cues in English) and treat each word as an atomic entity. This means that the model parameters $\boldsymbol{\theta}$ consist of the HMM state-to-state transition probabilities and the state-to-word emission probabilities.

In virtually all statistical approaches the parameters $\boldsymbol{\theta}$ are chosen or *estimated* on the basis of training data $\boldsymbol{d}$. This paper studies unsupervised estimation, so $\boldsymbol{d} = \boldsymbol{w} = (w_1, \ldots, w_n)$ consists of a sequence of words w_i containing all of the words of training corpus appended into a single string, as explained below.

Maximum Likelihood (ML) is the most common estimation method in computational linguistics. A Maximum Likelihood estimator sets the parameters to the value $\hat{\boldsymbol{\theta}}$ that makes the likelihood $L_{\boldsymbol{d}}$ of the data $\boldsymbol{d}$ as large as possible:

$$\begin{aligned} L_{\boldsymbol{d}}(\boldsymbol{\theta}) &= \mathrm{P}(\boldsymbol{d} \mid \boldsymbol{\theta}) \\ \hat{\boldsymbol{\theta}} &= \arg\max_{\theta} L_{\boldsymbol{d}}(\boldsymbol{\theta}) \end{aligned}$$

In this paper we use the Inside-Outside algorithm, which is a specialized form of Expectation-

Maximization, to find HMM parameters which (at least locally) maximize the likelihood function $L_{\boldsymbol{d}}$.

Recently there is increasing interest in Bayesian methods in computational linguistics, and the primary goal of this paper is to compare the performance of various Bayesian estimators with each other and with EM.

A Bayesian approach uses Bayes theorem to factorize the *posterior distribution* $\mathrm{P}(\boldsymbol{\theta} \mid \boldsymbol{d})$ into the *likelihood* $\mathrm{P}(\boldsymbol{d} \mid \boldsymbol{\theta})$ and the *prior* $\mathrm{P}(\boldsymbol{\theta})$.

$$\mathrm{P}(\boldsymbol{\theta} \mid \boldsymbol{d}) \quad \propto \quad \mathrm{P}(\boldsymbol{d} \mid \boldsymbol{\theta})\, \mathrm{P}(\boldsymbol{\theta})$$

Priors can be useful because they can express preferences for certain types of models. To take an example from our POS-tagging application, most words belong to relatively few parts-of-speech (e.g., most words belong to a single POS, and while there are some words which are both nouns and verbs, very few are prepositions and adjectives as well). One might express this using a prior which prefers HMMs in which the state-to-word emissions are *sparse*, i.e., each state emits few words. An appropriate Dirichlet prior can express this preference.

While it is possible to use Bayesian inference to find a single model, such as the Maximum A Posteriori or MAP value of $\boldsymbol{\theta}$ which maximizes the posterior $\mathrm{P}(\boldsymbol{\theta} \mid \boldsymbol{d})$, this is not necessarily the best approach (Bishop, 2006; MacKay, 2003). Instead, rather than commiting to a single value for the parameters $\boldsymbol{\theta}$ many Bayesians often prefer to work with the full posterior distribution $\mathrm{P}(\boldsymbol{\theta} \mid \boldsymbol{d})$, as this naturally reflects the uncertainty in $\boldsymbol{\theta}$'s value.

In all but the simplest models there is no known closed form for the posterior distribution. However, the Bayesian literature describes a number of methods for approximating the posterior $\mathrm{P}(\boldsymbol{\theta} \mid \boldsymbol{d})$. Monte Carlo sampling methods and Variational Bayes are two kinds of approximate inference methods that have been applied to Bayesian inference of unsupervised HMM POS taggers (Goldwater and Griffiths, 2007; Johnson, 2007). These methods can also be used to approximate other distributions that are important to us, such as the conditional distribution $\mathrm{P}(\boldsymbol{t} \mid \boldsymbol{w})$ of POS tags (i.e., HMM hidden states) $\boldsymbol{t}$ given words $\boldsymbol{w}$.

This recent literature reports contradictory results about these Bayesian inference methods. Johnson (2007) compared two Bayesian inference algorithms, Variational Bayes and what we call here a point-wise collapsed Gibbs sampler, and found that Variational Bayes produced the best solution, and that the Gibbs sampler was extremely slow to converge and produced a worse solution than EM. On the other hand, Goldwater and Griffiths (2007) reported that the same kind of Gibbs sampler produced much better results than EM on their unsupervised POS tagging task. One of the primary motivations for this paper was to understand and resolve the difference in these results. We replicate the results of both papers and show that the difference in their results stems from differences in the sizes of the training data and numbers of states in their models.

It turns out that the Gibbs sampler used in these earlier papers is not the only kind of sampler for HMMs. This paper compares the performance of four different kinds of Gibbs samplers, Variational Bayes and Expectation Maximization on unsupervised POS tagging problems of various sizes. Our goal here is to try to learn how the performance of these different estimators varies as we change the number of hidden states in the HMMs and the size of the training data.

In theory, the Gibbs samplers produce streams of samples that eventually converge on the true posterior distribution, while the Variational Bayes (VB) estimator only produces an approximation to the posterior. However, as the size of the training data distribution increases the likelihood function and therefore the posterior distribution becomes increasingly peaked, so one would expect this variational approximation to become increasingly accurate. Further the Gibbs samplers used in this paper should exhibit reduced mobility as the size of training data increases, so as the size of the training data increases eventually the Variational Bayes estimator should prove to be superior.

However the two point-wise Gibbs samplers investigated here, which resample the label of each word conditioned on the labels of its neighbours (amongst other things) only require $O(m)$ steps per sample (where m is the number of HMM states), while EM, VB and the sentence-blocked Gibbs samplers require $O(m^2)$ steps per sample. Thus for HMMs with many states it is possible to perform one or two orders of magnitude more iterations of the

point-wise Gibbs samplers in the same run-time as the other samplers, so it is plausible that they would yield better results.

2 Inference for HMMs

There are a number of excellent textbook presentations of Hidden Markov Models (Jelinek, 1997; Manning and Schütze, 1999), so we do not present them in detail here. Conceptually, a Hidden Markov Model uses a Markov model to generate the sequence of states $\boldsymbol{t} = (t_1, \ldots, t_n)$ (which will be interpreted as POS tags), and then generates each word w_i conditioned on the corresponding state t_i.

We insert endmarkers at the beginning and end of the corpus and between sentence boundaries, and constrain the estimators to associate endmarkers with a special HMM state that never appears elsewhere in the corpus (we ignore these endmarkers during evaluation). This means that we can formally treat the training corpus as one long string, yet each sentence can be processed independently by a first-order HMM.

In more detail, the HMM is specified by a pair of multinomials $\boldsymbol{\theta}_t$ and $\boldsymbol{\phi}_t$ associated with each state t, where $\boldsymbol{\theta}_t$ specifies the distribution over states t' following t and $\boldsymbol{\phi}_t$ specifies the distribution over words w given state t.

$$\begin{array}{rcrcl} t_i & | & t_{i-1} = t & \sim & \mathrm{Multi}(\boldsymbol{\theta}_t) \\ w_i & | & t_i = t & \sim & \mathrm{Multi}(\boldsymbol{\phi}_t) \end{array} \quad (1)$$

The Bayesian model we consider here puts a fixed uniform Dirichlet prior on these multinomials. Because Dirichlets are conjugate to multinomials, this greatly simplifies inference.

$$\begin{array}{rcrcl} \boldsymbol{\theta}_t & | & \alpha & \sim & \mathrm{Dir}(\boldsymbol{\alpha}) \\ \boldsymbol{\phi}_t & | & \alpha' & \sim & \mathrm{Dir}(\boldsymbol{\alpha}') \end{array}$$

A multinomial $\boldsymbol{\theta}$ is distributed according to the Dirichlet distribution $\mathrm{Dir}(\boldsymbol{\alpha})$ iff:

$$\mathrm{P}(\boldsymbol{\theta} \mid \boldsymbol{\alpha}) \;\propto\; \prod_{j=1}^{m} \theta_j^{\alpha_j - 1}$$

In our experiments we set $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}'$ to the uniform values (i.e., all components have the same value α or α'), but it is possible to estimate these as well (Goldwater and Griffiths, 2007). Informally, α controls the sparsity of the state-to-state transition probabilities while α' controls the sparsity of the state-to-word emission probabilities. As α' approaches zero the prior strongly prefers models in which each state emits as few words as possible, capturing the intuition that most word types only belong to one POS mentioned earlier.

2.1 Expectation Maximization

Expectation-Maximization is a procedure that iteratively re-estimates the model parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$, converging on a local maximum of the likelihood. Specifically, if the parameter estimate at iteration ℓ is $(\boldsymbol{\theta}^{(\ell)}, \boldsymbol{\phi}^{(\ell)})$, then the re-estimated parameters at iteration $\ell + 1$ are:

$$\begin{aligned} \theta_{t'|t}^{(\ell+1)} &= \mathrm{E}[n_{t',t}]/\mathrm{E}[n_t] \\ \phi_{w|t}^{(\ell+1)} &= \mathrm{E}[n'_{w,t}]/\mathrm{E}[n_t] \end{aligned} \quad (2)$$

where $n'_{w,t}$ is the number of times word w occurs with state t, $n_{t',t}$ is the number of times state t' follows t and n_t is the number of occurences of state t; all expectations are taken with respect to the model $(\boldsymbol{\theta}^{(\ell)}, \boldsymbol{\phi}^{(\ell)})$.

The experiments below used the Forward-Backward algorithm (Jelinek, 1997), which is a dynamic programming algorithm for calculating the likelihood and the expectations in (2) in $O(nm^2)$ time, where n is the number of words in the training corpus and m is the number of HMM states.

2.2 Variational Bayes

Variational Bayesian inference attempts to find a function $Q(\boldsymbol{t}, \boldsymbol{\theta}, \boldsymbol{\phi})$ that minimizes an upper bound (3) to the negative log likelihood.

$$\begin{aligned} & -\log \mathrm{P}(\boldsymbol{w}) \\ = \;& -\log \int Q(\boldsymbol{t}, \boldsymbol{\theta}, \boldsymbol{\phi}) \frac{\mathrm{P}(\boldsymbol{w}, \boldsymbol{t}, \boldsymbol{\theta}, \boldsymbol{\phi})}{Q(\boldsymbol{t}, \theta, \phi)} \, d\boldsymbol{t} \, d\boldsymbol{\theta} \, d\boldsymbol{\phi} \\ \leq \;& -\int Q(\boldsymbol{t}, \boldsymbol{\theta}, \boldsymbol{\phi}) \log \frac{\mathrm{P}(\boldsymbol{w}, \boldsymbol{t}, \boldsymbol{\theta}, \boldsymbol{\phi})}{Q(\boldsymbol{t}, \boldsymbol{\theta}, \boldsymbol{\phi})} \, d\boldsymbol{t} \, d\boldsymbol{\theta} \, d\boldsymbol{\phi} \end{aligned} \quad (3)$$

The upper bound (3) is called the *Variational Free Energy*. We make a "mean-field" assumption that the posterior can be well approximated by a factorized model Q in which the state sequence $\boldsymbol{t}$ does not covary with the model parameters $\boldsymbol{\theta}, \boldsymbol{\phi}$:

$$\mathrm{P}(\boldsymbol{t}, \boldsymbol{\theta}, \boldsymbol{\phi} \mid \boldsymbol{w}) \;\approx\; Q(\boldsymbol{t}, \boldsymbol{\theta}, \boldsymbol{\phi}) \;=\; Q_1(\boldsymbol{t}) Q_2(\boldsymbol{\theta}, \boldsymbol{\phi})$$

$$\mathrm{P}(t_i|\boldsymbol{w}, \boldsymbol{t}_{-i}, \alpha, \alpha') \quad \propto \quad \left(\frac{n'_{w_i,t_i} + \alpha'}{n_{t_i} + m'\alpha'}\right) \left(\frac{n_{t_i,t_{i-1}} + \alpha}{n_{t_{i-1}} + m\alpha}\right) \left(\frac{n_{t_{i+1},t_i} + \mathrm{I}(t_{i-1} = t_i = t_{i+1}) + \alpha}{n_{t_i} + \mathrm{I}(t_{i-1} = t_i) + m\alpha}\right)$$

Figure 1: The conditional distribution for state t_i used in the pointwise collapsed Gibbs sampler, which conditions on all states $\boldsymbol{t}_{-i}$ *except* t_i (i.e., the counts n do not include t_i). Here m' is the size of the vocabulary, m is the number of HMM states and $\mathrm{I}(\cdot)$ is the indicator function (i.e., equal to one if its argument is true and zero otherwise),

The calculus of variations is used to minimize the KL divergence between the desired posterior distribution and the factorized approximation. It turns out that if the likelihood and conjugate prior belong to exponential families then the optimal Q_1 and Q_2 do too, and there is an EM-like iterative procedure that finds locally-optimal model parameters (Bishop, 2006).

This procedure is especially attractive for HMM inference, since it involves only a minor modification to the M-step of the Forward-Backward algorithm. MacKay (1997) and Beal (2003) describe Variational Bayesian (VB) inference for HMMs. In general, the E-step for VB inference for HMMs is the same as in EM, while the M-step is as follows:

$$\begin{aligned} \tilde{\theta}^{(\ell+1)}_{t'|t} &= f(\mathrm{E}[n_{t',t}] + \alpha)/f(\mathrm{E}[n_t] + m\alpha) \quad (4) \\ \tilde{\phi}^{(\ell+1)}_{w|t} &= f(\mathrm{E}[n'_{w,t}] + \alpha')/f(\mathrm{E}[n_t] + m'\alpha') \\ f(v) &= \exp(\Psi(v)) \end{aligned}$$

where m' and m are the number of word types and states respectively, Ψ is the digamma function and the remaining quantities are as in (2). This means that a single iteration can be performed in $O(nm^2)$ time, just as for the EM algorithm.

2.3 MCMC sampling algorithms

The goal of Markov Chain Monte Carlo (MCMC) algorithms is to produce a stream of samples from the posterior distribution $\mathrm{P}(\boldsymbol{t} \mid \boldsymbol{w}, \boldsymbol{\alpha})$. Besag (2004) provides a tutorial on MCMC techniques for HMM inference.

A Gibbs sampler is a simple kind of MCMC algorithm that is well-suited to sampling high-dimensional spaces. A Gibbs sampler for $\mathrm{P}(\boldsymbol{z})$ where $\boldsymbol{z} = (z_1, \ldots, z_n)$ proceeds by sampling and updating each z_i in turn from $\mathrm{P}(z_i \mid \boldsymbol{z}_{-i})$, where $\boldsymbol{z}_{-i} = (z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n)$, i.e., all of the z *except* z_i (Geman and Geman, 1984; Robert and Casella, 2004).

We evaluate four different Gibbs samplers in this paper, which vary along two dimensions. First, the sampler can either be *pointwise* or *blocked*. A pointwise sampler resamples a single state t_i (labeling a single word w_i) at each step, while a blocked sampler resamples the labels for all of the words in a sentence at a single step using a dynamic programming algorithm based on the Forward-Backward algorithm. (In principle it is possible to use block sizes other than the sentence, but we did not explore this here). A pointwise sampler requires $O(nm)$ time per iteration, while a blocked sampler requires $O(nm^2)$ time per iteration, where m is the number of HMM states and n is the length of the training corpus.

Second, the sampler can either be *explicit* or *collapsed*. An explicit sampler represents and samples the HMM parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ in addition to the states $\boldsymbol{t}$, while in a collapsed sampler the HMM parameters are integrated out, and only the states $\boldsymbol{t}$ are sampled. The difference between explicit and collapsed samplers corresponds exactly to the difference between the two PCFG sampling algorithms presented in Johnson et al. (2007).

An iteration of the pointwise explicit Gibbs sampler consists of resampling $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ given the state-to-state transition counts $\boldsymbol{n}$ and state-to-word emission counts $\boldsymbol{n}'$ using (5), and then resampling each state t_i given the corresponding word w_i and the neighboring states t_{i-1} and t_{i+1} using (6).

$$\begin{aligned} \boldsymbol{\theta}_t \mid \boldsymbol{n}_t, \boldsymbol{\alpha} &\sim \mathrm{Dir}(\boldsymbol{n}_t + \boldsymbol{\alpha}) \\ \boldsymbol{\phi}_t \mid \boldsymbol{n}'_t, \boldsymbol{\alpha}' &\sim \mathrm{Dir}(\boldsymbol{n}'_t + \boldsymbol{\alpha}') \quad (5) \\ \mathrm{P}(t_i \mid w_i, \boldsymbol{t}_{-i}, \boldsymbol{\theta}, \boldsymbol{\phi}) &\propto \theta_{t_i|t_{i-1}} \phi_{w_i|t_i} \theta_{t_{i+1}|t_i} \quad (6) \end{aligned}$$

The Dirichlet distributions in (5) are non-uniform; $\boldsymbol{n}_t$ is the vector of state-to-state transition counts in $\boldsymbol{t}$ leaving state t in the current state vector $\boldsymbol{t}$, while

$\boldsymbol{n}'_t$ is the vector of state-to-word emission counts for state t. See Johnson et al. (2007) for a more detailed explanation, as well as an algorithm for sampling from the Dirichlet distributions in (5).

The samplers that Goldwater and Griffiths (2007) and Johnson (2007) describe are pointwise collapsed Gibbs samplers. Figure 1 gives the sampling distribution for this sampler. As Johnson et al. (2007) explains, samples of the HMM parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ can be obtained using (5) if required.

The blocked Gibbs samplers differ from the pointwise Gibbs samplers in that they resample the POS tags for an entire sentence at a time. Besag (2004) describes the well-known dynamic programming algorithm (based on the Forward-Backward algorithm) for sampling a state sequence $\boldsymbol{t}$ given the words $\boldsymbol{w}$ and the transition and emission probabilities $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$.

At each iteration the explicit blocked Gibbs sampler resamples $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ using (5), just as the explicit pointwise sampler does. Then it uses the new HMM parameters to resample the states $\boldsymbol{t}$ for the training corpus using the algorithm just mentioned. This can be done in parallel for each sentence in the training corpus.

The collapsed blocked Gibbs sampler is a straight-forward application of the Metropolis-within-Gibbs approach proposed by Johnson et al. (2007) for PCFGs, so we only sketch it here. We iterate through the sentences of the training data, resampling the states for each sentence conditioned on the state-to-state transition counts $\boldsymbol{n}$ and state-to-word emission counts $\boldsymbol{n}'$ for the other sentences in the corpus. This is done by first computing the parameters $\boldsymbol{\theta}^\star$ and $\boldsymbol{\phi}^\star$ of a *proposal HMM* using (7).

$$\begin{aligned} \theta^\star_{t'|t} &= \frac{n_{t',t} + \alpha}{n_t + m\alpha} \\ \phi^\star_{w|t} &= \frac{n'_{w,t} + \alpha'}{n_t + m'\alpha} \end{aligned} \tag{7}$$

Then we use the dynamic programming sampler described above to produce a *proposal state sequence* $\boldsymbol{t}^\star$ for the words in the sentence. Finally, we use a Metropolis-Hastings accept-reject step to decide whether to update the current state sequence for the sentence with the proposal $\boldsymbol{t}^\star$, or whether to keep the current state sequence. In practice, with all but the very smallest training corpora the acceptance rate is very high; the acceptance rate for all of our collapsed blocked Gibbs samplers was over 99%.

3 Evaluation

The previous section described six different unsupervised estimators for HMMs. In this section we compare their performance for English part-of-speech tagging. One of the difficulties in evaluating unsupervised taggers such as these is mapping the system's states to the gold-standard parts-of-speech. Goldwater and Griffiths (2007) proposed an information-theoretic measure known as the *Variation of Information* (VI) described by Meilă (2003) as an evaluation of an unsupervised tagging. However as Goldwater (p.c.) points out, this may not be an ideal evaluation measure; e.g., a tagger which assigns all words the same single part-of-speech tag does disturbingly well under Variation of Information, suggesting that a poor tagger may score well under VI.

In order to avoid this problem we focus here on evaluation measures that construct an explicit mapping between the gold-standard part-of-speech tags and the HMM's states. Perhaps the most straightforward approach is to map each HMM state to the part-of-speech tag it co-occurs with most frequently, and use this mapping to map each HMM state sequence $\boldsymbol{t}$ to a sequence of part-of-speech tags. But as Clark (2003) observes, this approach has several defects. If a system is permitted to posit an unbounded number of states (which is not the case here) it can achieve a perfect score on by assigning each word token its own unique state.

We can partially address this by cross-validation. We divide the corpus into two equal parts, and from the first part we extract a mapping from HMM states to the parts-of-speech they co-occur with most frequently, and use that mapping to map the states of the second part of the corpus to parts-of-speech. We call the accuracy of the resulting tagging the *cross-validation accuracy*.

Finally, following Haghighi and Klein (2006) and Johnson (2007) we can instead insist that at most one HMM state can be mapped to any part-of-speech tag. Following these authors, we used a greedy algorithm to associate states with POS tags; the accuracy of the resulting tagging is called the *greedy 1-to-1*

	All − 50	All − 17	120K − 50	120K − 17	24K − 50	24K − 17
EM	0.40527	0.43101	0.29303	0.35202	0.18618	0.28165
VB	0.46123	**0.51379**	0.34679	0.36010	0.23823	0.36599
$GS_{e,p}$	0.47826	0.43424	0.36984	0.44125	0.29953	0.36811
$GS_{e,b}$	0.49371	0.46568	0.38888	**0.44341**	0.34404	0.37032
$GS_{c,p}$	**0.49910***	0.45028	**0.42785**	0.43652	**0.39182**	**0.39164**
$GS_{c,b}$	0.49486*	0.46193	0.41162	0.42278	0.38497	0.36793

Figure 2: Average greedy 1-to-1 accuracy of state sequences produced by HMMs estimated by the various estimators. The column heading indicates the size of the corpus and the number of HMM states. In the Gibbs sampler (GS) results the subscript "e" indicates that the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ were explicitly sampled while the subscript "c" indicates that they were integrated out, and the subscript "p" indicates pointwise sampling, while "b" indicates sentence-blocked sampling. Entries tagged with a star indicate that the estimator had not converged after weeks of run-time, but was still slowly improving.

	All − 50	All − 17	120K − 50	120K − 17	24K − 50	24K − 17
EM	0.62115	0.64651	0.44135	0.56215	0.28576	0.46669
VB	0.60484	0.63652	0.48427	0.36458	0.35946	0.36926
$GS_{e,p}$	0.64190	0.63057	0.53571	0.46986	0.41620	0.37165
$GS_{e,b}$	**0.65953**	0.65606	0.57918	0.48975	0.47228	0.37311
$GS_{c,p}$	0.61391*	**0.67414**	**0.65285**	**0.65012**	**0.58153**	**0.62254**
$GS_{c,b}$	0.60551*	0.65516	0.62167	0.58271	0.55006	0.58728

Figure 3: Average cross-validation accuracy of state sequences produced by HMMs estimated by the various estimators. The table headings follow those used in Figure 2.

	All − 50	All − 17	120K − 50	120K − 17	24K − 50	24K − 17
EM	4.47555	3.86326	6.16499	4.55681	7.72465	5.42815
VB	4.27911	3.44029	5.00509	3.19670	4.80778	3.14557
$GS_{e,p}$	4.24919	3.53024	4.30457	3.23082	**4.24368**	3.17076
$GS_{e,b}$	4.04123	**3.46179**	4.22590	3.20276	4.29474	**3.10609**
$GS_{c,p}$	**4.03886***	3.52185	**4.21259**	**3.17586**	4.30928	3.18273
$GS_{c,b}$	4.11272*	3.61516	4.36595	3.23630	4.32096	3.17780

Figure 4: Average Variation of Information between the state sequences produced by HMMs estimated by the various estimators and the gold tags (smaller is better). The table headings follow those used in Figure 2.

	All − 50	All − 17	120K − 50	120K − 17	24K − 50	24K − 17
EM	558	346	648	351	**142**	125
VB	**473**	**123**	**337**	**24**	183	**20**
$GS_{e,p}$	2863	382	3709	63	2500	177
$GS_{e,b}$	3846	286	5169	154	4856	139
$GS_{c,p}$	*	34325	44864	40088	45285	43208
$GS_{c,b}$	*	6948	7502	7782	7342	7985

Figure 5: Average number of iterations until the negative logarithm of the posterior probability (or likelihood) changes by less than 0.5% (smaller is better) per at least 2,000 iterations. No annealing was used.

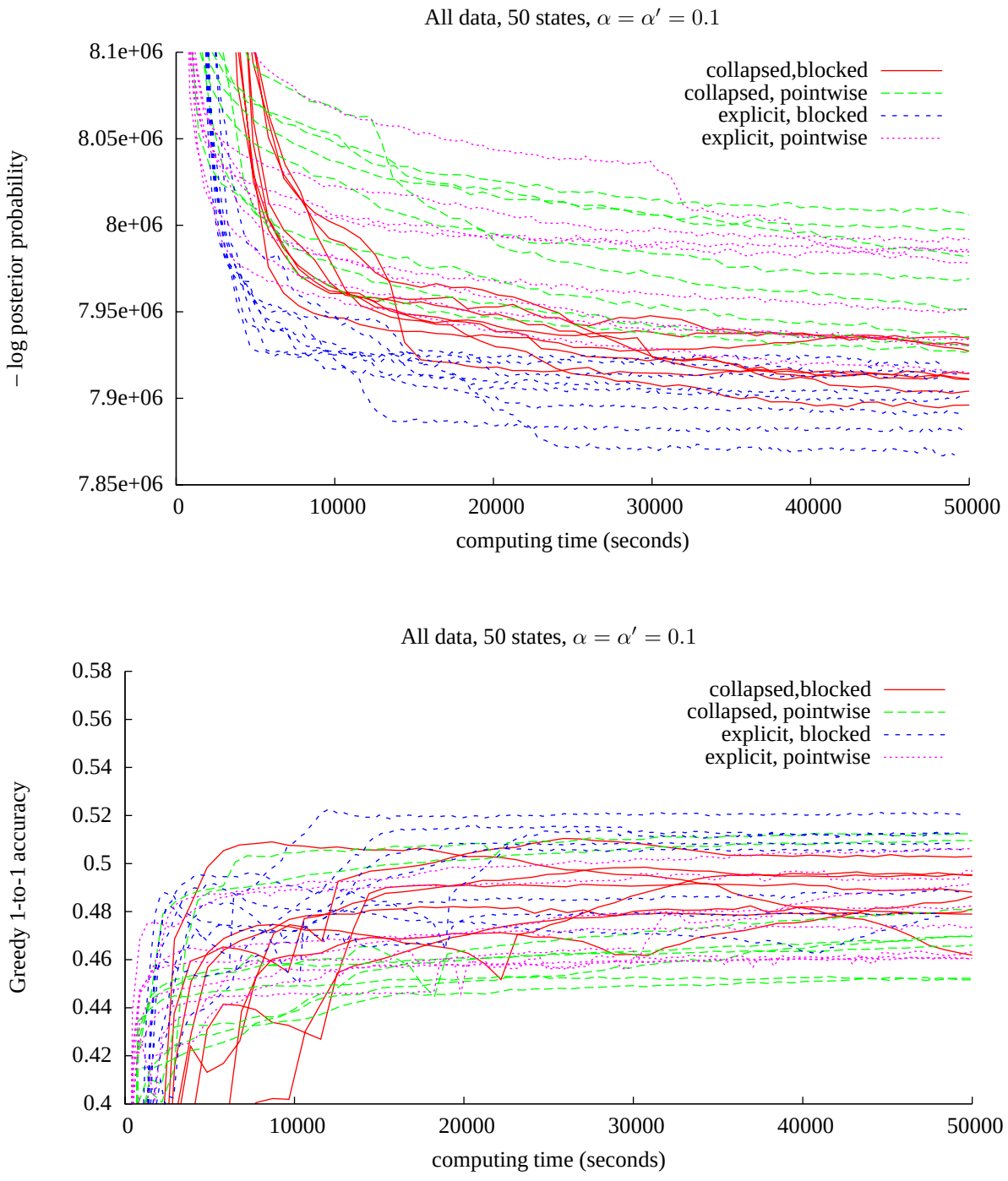

Figure 6: Variation in (a) negative log likelihood and (b) 1-to-1 accuracy as a function of running time on a 3GHz dual quad-core Pentium for the four different Gibbs samplers on all data and 50 hidden states. Each iteration took approximately 96 sec. for the collapsed blocked sampler, 7.5 sec. for the collapsed pointwise sampler, 25 sec. for the explicit blocked sampler and 4.4 sec. for the explicit pointwise sampler.

accuracy.

The studies presented by Goldwater and Griffiths (2007) and Johnson (2007) differed in the number of states that they used. Goldwater and Griffiths (2007) evaluated against the reduced tag set of 17 tags developed by Smith and Eisner (2005), while Johnson (2007) evaluated against the full Penn Treebank tag set. We ran all our estimators in both conditions here (thanks to Noah Smith for supplying us with his tag set).

Also, the studies differed in the size of the corpora used. The largest corpus that Goldwater and Griffiths (2007) studied contained 96,000 words, while Johnson (2007) used all of the 1,173,766 words in the full Penn WSJ treebank. For that reason we ran all our estimators on corpora containing 24,000 words and 120,000 words as well as the full treebank.

We ran each estimator with the eight different combinations of values for the hyperparameters α and α' listed below, which include the optimal values for the hyperparameters found by Johnson (2007), and report results for the best combination for each estimator below [1].

α	α'
1	1
1	0.5
0.5	1
0.5	0.5
0.1	0.1
0.1	0.0001
0.0001	0.1
0.0001	0.0001

Further, we ran each setting of each estimator at least 10 times (from randomly jittered initial starting points) for at least 1,000 iterations, as Johnson (2007) showed that some estimators require many iterations to converge. The results of our experiments are summarized in Figures 2–5.

[1] We found that on some data sets the results are sensitive to the values of the hyperparameters. So, there is a bit uncertainty in our comparison results because it is possible that the values we tried were good for one estimator and bad for others. Unfortunately, we do not know any efficient way of searching the optimal hyperparameters in a much wider and more fine-grained space. We leave it to future work.

4 Conclusion and future work

As might be expected, our evaluation measures disagree somewhat, but the following broad tendancies seem clear. On small data sets all of the Bayesian estimators strongly outperform EM (and, to a lesser extent, VB) with respect to all of our evaluation measures, confirming the results reported in Goldwater and Griffiths (2007). This is perhaps not too surprising, as the Bayesian prior plays a comparatively stronger role with a smaller training corpus (which makes the likelihood term smaller) and the approximation used by Variational Bayes is likely to be less accurate on smaller data sets.

But on larger data sets, which Goldwater et al did not study, the results are much less clear, and depend on which evaluation measure is used. Expectation Maximization does surprisingly well on larger data sets and is competitive with the Bayesian estimators at least in terms of cross-validation accuracy, confirming the results reported by Johnson (2007).

Variational Bayes converges faster than all of the other estimators we examined here. We found that the speed of convergence of our samplers depends to a large degree upon the values of the hyperparameters α and α', with larger values leading to much faster convergence. This is not surprising, as the α and α' specify how likely the samplers are to consider novel tags, and therefore directly influence the sampler's mobility. However, in our experiments the best results are obtained in most settings with small values for α and α', usually between 0.1 and 0.0001.

In terms of time to convergence, on larger data sets we found that the blocked samplers were generally faster than the pointwise samplers, and that the explicit samplers (which represented and sampled $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$) were faster than the collapsed samplers, largely because the time saved in not computing probabilities on the fly overwhelmed the time spent resampling the parameters.

Of course these experiments only scratch the surface of what is possible. Figure 6 shows that pointwise-samplers initially converge faster, but are overtaken later by the blocked samplers. Inspired by this, one can devise hybrid strategies that interleave blocked and pointwise sampling; these might perform better than both the blocked and pointwise samplers described here.

References

Matthew J. Beal. 2003. *Variational Algorithms for Approximate Bayesian Inference*. Ph.D. thesis, Gatsby Computational Neuroscience unit, University College London.

Julian Besag. 2004. An introduction to Markov Chain Monte Carlo methods. In Mark Johnson, Sanjeev P. Khudanpur, Mari Ostendorf, and Roni Rosenfeld, editors, *Mathematical Foundations of Speech and Language Processing*, pages 247–270. Springer, New York.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

Alexander Clark. 2003. Combining distributional and morphological information for part of speech induction. In *10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 59–66. 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.

Sharon Goldwater and Tom Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 744–751, Prague, Czech Republic, June. Association for Computational Linguistics.

Aria Haghighi and Dan Klein. 2006. Prototype-driven learning for sequence models. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 320–327, New York City, USA, June. Association for Computational Linguistics.

Frederick Jelinek. 1997. *Statistical Methods for Speech Recognition*. The MIT Press, Cambridge, Massachusetts.

Mark Johnson, Thomas Griffiths, and Sharon Goldwater. 2007. 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. 2007. Why doesn't EM find good HMM POS-taggers? In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 296–305.

David J.C. MacKay. 1997. Ensemble learning for hidden Markov models. Technical report, Cavendish Laboratory, Cambridge.

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

Chris Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts.

Marina Meilă. 2003. Comparing clusterings by the variation of information. In Bernhard Schölkopf and Manfred K. Warmuth, editors, *COLT 2003: The Sixteenth Annual Conference on Learning Theory*, volume 2777 of *Lecture Notes in Computer Science*, pages 173–187. Springer.

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

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 354–362, Ann Arbor, Michigan, June. Association for Computational Linguistics.