A comparison of unsupervised methods for Part-of-Speech Tagging in Chinese

Alex Cheng
Microsoft Corporation
alcheng@microsoft.com

Fei Xia
Univ. of Washington
fxia@uw.edu

Jianfeng Gao
Microsoft Research
jfgao@microsoft.com

Abstract

We conduct a series of Part-of-Speech (POS) Tagging experiments using Expectation Maximization (EM), Variational Bayes (VB) and Gibbs Sampling (GS) against the Chinese Penn Treebank. We want to first establish a baseline for unsupervised POS tagging in Chinese, which will facilitate future research in this area. Secondly, by comparing and analyzing the results between Chinese and English, we highlight some of the strengths and weaknesses of each of the algorithms in POS tagging task and attempt to explain the differences based on some preliminary linguistics analysis. Comparing to English, we find that all algorithms perform rather poorly in Chinese in 1-to-1 accuracy result but are more competitive in many-to-1 accuracy. We attribute one possible explanation of this to the algorithms' inability to correctly produce tags that match the desired tag count distribution.

1 Introduction

Recently, there has been much work on unsupervised POS tagging using Hidden Markov Models (Johnson, 2007; Goldwater & Griffiths, 2007). Three common approaches are Expectation Maximization (EM), Variational Bayes (VB) and Gibbs Sampling (GS). EM was first used in POS tagging in (Merialdo, 1994) which showed that except in conditions where there are no labeled training data at all, EM performs very poorly. Gao and Johnson (2008) compared EM, VB and GS in English against the Penn Treebank Wall Street Journal (WSJ) text. Their experiments on English showed that GS outperforms EM and VB in almost all cases. Other notable studies in the unsupervised and semi-supervised POS domain include the use of prototype examples (Haghighi & Klien, 2006), dictionary constraints to guide the algorithms (Elworthy 1994; Banko & Moore 2004) and Bayseian LDA-based model (Toutanova and Johnson, 2007).

To our knowledge, little work has been done on unsupervised POS tagging in Chinese against the Chinese Penn Treebank (CTB). The work in Chinese POS tagging has been predominately in the supervised fashion (Huang et al. 2009; Chang & Chen, 1993; Ng & Low, 2004) and achieve accuracy of 92.25% using a traditional ngram HMM tagger. For English, a supervised trigram tagger achieves an accuracy of 96.7% against the Penn Treebank (Thorsten, 2000).

In this study, we analyze and compare the performance of three classes of unsupervised learning algorithms on Chinese and report the experimental results on the CTB. We establish a baseline for unsupervised POS tagging in Chinese. We then compare and analyze the results between Chinese and English, we explore some of the strengths and weaknesses of each of the algorithms in POS tagging task and attempt to explain the differences based on some preliminary linguistics analysis.

2 Models

In this section, we provide a brief overview of the three unsupervised learning methods for POS tagging as described in (Gao & Johnson, 2008), which all uses a traditional bigram Hidden Markov Model (HMM). HMM is a well-known statistical model, used for sequential modeling. To put it formally, let $T = \{t_1, \dots, t_i, \dots, t_j, \dots, t_{|T|}\}$ be the set of possible states and $W = \{w_1, \dots, w_k, \dots, w_{|W|}\}$ be the set of possible observations. In the case for POS tagging using a bigram model, the set T corresponds to the set of POS tags and the set W corresponds to the set of words in the language.

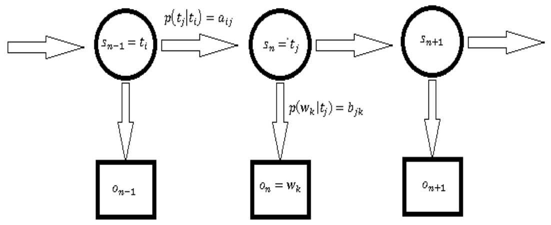

Figure 1: Graphical model of an HMM for a bigram POS tagger. The top row represents a sequence of hidden states where each is conditionally dependent only on the previous state and the bottom row represents a sequence of observations where each is conditionally dependent only on the current state.

An HMM models a sequence of discrete observations $o_{1:N} = \{o_1, \dots, o_N\}$ where $o_n = w_k$ that are produced by a sequence of hidden states $\boldsymbol{s}_{1:N} = \{s_1, \dots, s_N\}$ where $s_n = t_i$. The sequence of states is produced by a first order Markov process such that the current state s_n depends only on its previous state s_{n-1}; correspondingly each of the observations o_n depends only on the state s_n:

$$\begin{aligned} p(s_n|\boldsymbol{s_{1:n-1}}) &\cong p(s_n|s_{n-1}) \\ p(o_n|\boldsymbol{s_{1:n}}\, \boldsymbol{o}_{1:n-1}) &\cong p(o_n|s_n) \end{aligned} \qquad (1)$$

where $a_{ij} = p(s_n|s_{n-1})$ is the probability of transition to state $s_n = t_j$ from $s_{n-1} = t_i$ and $b_{jk} = p(o_n|s_n)$ is the probability of observation $o_n = w_k$ produced by $s_n = t_j$. The parameter θ for the HMM is defined by the transition probability distribution $A \equiv (a_{ij})$, emission (observation) probability distribution $B \equiv (b_{jk})$ and the initial probability $\pi \equiv \prod p(s_0 = t_i)$. Direct calculation of the likelihood $p(s, o|\theta)$ is computationally inefficient, and we can use dynamic programming techniques to speed up the calculation by calculating the forward probability:

$$\alpha_i(n) \equiv p(o_{1:n}, s_n = t_i|\theta) \qquad (2)$$

and backward probability

$$\beta_i(n) \equiv p(o_{n:N}|s_n = t_i, \theta). \qquad (3)$$

See (Mannings & Schutze, 1999) for details on the calculation.

2.1 Expectation Maximization (EM)

EM is a general class of algorithms for finding the maximum likelihood estimator of parameters in probabilistic models. It is an iterative algorithm where we alternate between calculating the expectation of the log likelihood of the model given the parameters:

$$Q(\theta|\theta^t) = \mathbb{E}_{p(s|o,\theta)} \ln p(s, o|\theta) \qquad (4)$$

and then finding the parameters that maximizes the expected log likelihood. Using Lagrange multipliers with constraint that each parameter is a probability distribution, we have these update steps for the well-known forward-backward Algorithm for EM HMM:

$$\pi^{t+1} : \pi_i = \alpha_i(1)\beta_i(1)$$

$$A^{t+1} : a_{ij} = \frac{\sum_{n=2} \alpha_i(n) a_{ij} b_{j o_n} \beta_j(n+1)}{\sum_{i=1}^{T} \alpha_i(n) \beta_i(n)} \qquad (5)$$

$$B^{t+1} : b_{jk} = \sum_{n=1}^{N} \frac{\alpha_i(n) a_{ij} b_{j o_n} \beta_j(n+1) \delta(o_n, k)}{\sum_{i=1}^{T} \alpha_i(n) \beta_i(n)}$$

where $\delta(o_n, k) = \begin{cases} 1, & o_n = w_k \\ 0, & o.w. \end{cases}$.

2.2 Variational Bayes (VB)

One of the drawbacks of EM is that the resulting distribution is very uniform; that is, EM applies roughly the same number of observations for each state. Instead of using only the best model for decoding, the Bayesian approach uses and considers all the models; that is, the model is treated as a hidden variable. This is done by assigning a probability distribution over the model parameters as a prior distribution, $p(\theta)$.

In HMM, we calculate the probability of the observation by considering all models and integrating over the distribution over the priors:

$$p(\boldsymbol{o}_{1:N}) = \int p(o_{1:N}|\theta)p(\theta)\,d\theta \tag{6}$$
where $p(\theta) = p(\pi)p(A)p(B)$.

As with the standard in the literature, we use Dirichlet Prior as it allows us to model the tag distribution more closely and because they are in the same conjugate exponential family as the log likelihood. The Dirichlet distribution is parameterized by a vector of real values $\boldsymbol{\alpha}$ (hyper-parameters). There are two ways that we can view the vector $\boldsymbol{\alpha}$. First, the parameter controls the sharpness of distribution for each of the components. This is in contrast to the EM model where we essentially have a uniform prior. Thus, we can view $\boldsymbol{\alpha}$ as our prior beliefs on the shape of the distribution and we can make our choices based on our linguistics knowledge. Second, we can view the role of $\boldsymbol{\alpha}$ in terms of predictive distribution based on the statistics from observed counts. For HMM, we can set a separate prior for each state-state transition and word-state emission distribution, effectively giving us control over the distribution of each entry in the transition matrix. However, to simplify the model and without the need to fine tune each parameters, we use two fixed hyper-parameters: all of the state-state probability will have the hyper-parameter α_{TT} and all of the word-state probability will have hyper-parameter α_{WT}.

To begin our estimation and maximization procedure, we create $q(\theta, s_{1:N}) \approx q(\theta)q(s_{1:N})$ as an approximation of the posterior of the log likelihood:

$$\ln p(\boldsymbol{o}_{1:N}) \geq \int \sum_{s} q(\theta, s_{1:N}) \ln \frac{p(o_{1:N}, s_{1:N}|\theta)p(\theta)}{q(\theta, s_{1:N})}\,d\theta \tag{7}$$

By taking the functional derivative with respect to $q(\theta)$ to find the distribution that maximizes the log likelihood, and following the derivation from (Beal, 2003), we arrive at the following EM-like procedure:

$$\begin{aligned} &\mathrm{E}_{q(s)}[\delta(s_1, t_i)] \\ &= \alpha_i(1)\beta_i(1)\mathrm{E}_{q(s)}\left[\delta(s_{n-1}, t_i)\delta(s_n, t_j)\right] \\ &= \frac{\alpha_i(n)a_{ij}b_{jo_n}\beta_j(n+1)}{\sum_{i=1}^{T}\alpha_i(n)\beta_i(n)} \end{aligned} \tag{8}$$

$$\begin{aligned} &\mathrm{E}_{q(s)}[\delta(s_n, t_i)\delta(o_n, w_k)] \\ &= \frac{\alpha_i(n)a_{ij}b_{jo_n}\beta_j(n+1)\delta(o_n, w_k)}{\sum_{i=1}^{T}\alpha_i(n)\beta_i(n)} \end{aligned}$$

This is the Expectation step where α and β is the forward and backward probabilities and $\delta(o_n, w_k)$ is the indicator function as in EM.

The Maximization step is as follows:

$$\begin{aligned} &\pi^{t+1} : \pi_i \\ &= \frac{\exp(\psi(\alpha_{TT} + \mathrm{E}_{q(s)}[\delta(s_1, t_i)]))}{\exp(\psi(\sum_{i=1}^{T}\alpha_{TT} + \mathrm{E}_{q(s)}[\delta(s_1, t_i)]))} \\ &A^{t+1} : a_{ij} \\ &= \frac{\exp(\psi(\alpha_{TT} + \sum_{n=2}^{N}\mathrm{E}_{q(s)}[*]))}{\exp(\psi(\sum_{i=1}^{T}\alpha_{TT} + \sum_{n=2}^{N}\mathrm{E}_{q(s)}[*]))} \\ &B^{t+1} : b_{jk} \\ &= \frac{\exp(\psi(\alpha_{WT} + \sum_{n=1}^{N}\mathrm{E}_{q(s)}[**]))}{\exp(\psi(\sum_{i=1}^{T}\alpha_{WT} + \sum_{n=1}^{N}\mathrm{E}_{q(s)}[**]))} \end{aligned} \tag{9}$$

where $\mathrm{E}_{q(s)}[*] = E_{q(s)}[\delta(s_{n-1}, t_i)\delta(s_n, t_j)]$, $\mathrm{E}_{q(s)}[**] = E_{q(s)}[\delta(s_n, t_j)\delta(o_n, w_k)]$ and ψ is the digamma function.

2.3 Gibbs Sampling (GS)

Gibbs sampling (Geman & Geman, 1984) is a widely used MCMC algorithm designed especially for cases where we can sample from the conditional probability easily. It is a straightforward application of the Metropolis Hasting algorithm where we sample a variable t_k while keeping $t_{\backslash k}$ constant where $t_{\backslash k} = \{t_1, \ldots, t_{k-1}, t_{k+1}, \ldots, t_T\}$. We set the proposal distribution to

$$q_k(t^*|t^n) = p(t_k|t_{\backslash k}). \tag{10}$$

So the sampling procedure is the following: initialize the components of $\boldsymbol{t} = \{t_1,, \ldots, t_T\}$. Then sample t_1 from $p(t_1|t_2, \ldots, t_T)$, t_2 from $p(t_2|t_1, t_3, \ldots, t_T)$, and so on for each component of t. For POS tagging, the main idea is that we sample the tag t based on the $p(t|t')$ and $p(w|t)$ distribution.

The main idea for using GS for POS tagging is that in each iteration, we sample the tag t based on the $p(t|t')$ and $p(w|t)$ distribution.

Then from the samples, we count the number for each state-state and word-state pairs and update the probabilities accordingly. How we sample the data depends on whether we are using word based or sentence based sampling (the Expectation Step). Whereas how we update the probabilities depend on whether we are using a collapsed or explicit Gibbs sampler (the Maximization Step).

Word Based vs. Sentence Based: Word-based and sentence-based approaches to GS determine how we sample the each tag t at position n in the data set. For the word-based approach, instead of going through sentence by sentence (as in EM and VB procedures), we pick a word position in the corpus at random (without repetition) and sample a new tag t_i at position n using the probability:

$$p(s_n = t_i) = p(t_i|s_{n-1})p(w_k = o_n|t_i)p(s_{n+1}|t_i) \tag{11}$$

Notice that since we are selecting each position at random, the tag s_{n-1} at position n-1 and s_{n+1} at position n+1 are our samples at the previous iteration or an already updated samples at the current iteration.

The sentence-based approach use the forward and backward probability to sample the tag based on the sentence (Besag, 2004). Specifically, we use the backward probability $\beta_i(n) \equiv p(o_{1:n}|s_n = t_i, \theta)$ to sample the sentence from start ($n = 1$) to finish ($n = N$). We sample a new tag t_i at position n using the probability:

$$p(s_n = t_i|o_{1:n}, s_{1:n-1}, \theta) = p(t_i|s_{n-1})p(w_k = o_n|t_i)\beta_i(n) \tag{12}$$

where the transition and emission probability distribution are from the current model parameters. Again s_{n-1} is our "guess" at the previous sampling step of the tag of s_{n-1}.

Explicit vs. Collapsed Based: We use the tags estimated at the previous step to maximize the parameters. Our choice of using Dirichlet distributions over the parameters $P(A)$ and $P(B)$ give us some nice mathematical properties. We show that $p(A|\boldsymbol{o}_{1:N})$ and $p(B|\boldsymbol{o}_{1:N})$ also calculate to be Dirichlet distributions. Following (MacKay & Peto, 1994), the posterior probability of A can be derived as follows:

$$\begin{aligned} p(A|\boldsymbol{o}_{1:N}) &= \frac{p(\boldsymbol{o}_{1:N}|A)p(A)}{p(\boldsymbol{o}_{1:N})} \\ &= \frac{1}{p(\boldsymbol{o}_{1:N})}\prod_{n=1}^{N} a_{s_n,s_{n-1}} \prod_{i}^{T}\prod_{j}^{T} \frac{a_{ij}^{\alpha-1}\Gamma(\alpha_{\mathrm{TT}})}{\Gamma(|T|\alpha_{TT})} \\ &= \prod_{i=1}^{T}\prod_{j=1}^{T} \frac{a_{ij}^{C(t_i,t_j)+\alpha_{TT}-1}\Gamma\left(\alpha_{TT} + C(t_i,t_j)\right)}{\Gamma(|T|\alpha_{TT} + \sum c(t_i,t_j))} \\ &= \prod_{j} Dir\left(a_{ij}|\ c(t_i,t_j) + \alpha_{TT}\right) \end{aligned} \tag{13}$$

where $c(t_i, t_j)$ is the number of times t_i is followed by t_j in the sample from the previous iteration.

Similarly, we can define $p(B|o_{1:N})$ using the count $c(w_k, t_j)$ to show that:

$$p(B|\boldsymbol{o}_{1:N}) = \prod_{k} Dir\left(b_{jk}|\ c(w_k,t_j) + \alpha_{WT}\right) \tag{14}$$

For the collapsed Gibbs sampler, we want to integrate over all possible model parameters A to maximize the new transition probabilities using Maximum a posteriori (MAP) estimator:

$$\begin{aligned} p(t_j|t_i, o_{1:N}) &= \int p(t_j|t_i, A, o_{1:N})p(A|\boldsymbol{o}_{1:N})dA \\ &= \int a_{ij} \prod_{j} Dir\left(a_{ij}|\ c(t_i,t_j) + \alpha_{TT}\right) dA \\ &= \frac{c(t_i,t_j) + \alpha_{TT}}{\sum c(t_i,t_j) + |T|\alpha_{TT}} \end{aligned} \tag{15}$$

The last equality uses the following result:

$$\int \boldsymbol{\pi} Dir(\boldsymbol{\pi}|u)d\boldsymbol{\pi} = \frac{u}{|\pi|u} \tag{16}$$

We can derive a similar result for $p(w_k|t_i, o_{1:N})$. Then we can use the sample count to update the new parameter values.

An explicit sampler samples the HMM parameters θ in addition to the states. Specifically, in the Bayesian model, we will need to sample from the Dirichlet distribution for the parameters

$$p(A|o_{1:N}) = \prod_j Dir\big(a_{ij}|\, c(t_i,t_j) + \alpha_{TT}\big) \tag{17}$$

$$p(B|\boldsymbol{o}_{1:N}) = \prod_k Dir\big(b_{jk}|\, c(t_j,w_k) + \alpha_{WT}\big)$$

derived above. An n-dimensional Dirichlet distribution variable can be generated from gamma variate (Wolfram Mathematica, 2009):

$$u_{ij} \sim f\big(x, c(t_i,t_j) + \alpha_{TT}\big) = \frac{x^{c(t_i,t_j)+\alpha_{TT}-1} e^{-x}}{\Gamma\big(c(t_i,t_j) + \alpha_{TT}\big)} \tag{18}$$

we can update the transition probability by generating the gamma variate for the Dirichlet distribution:

$$a_{ij} \leftarrow \frac{u_{ij}}{\sum_j u_{ij}}. \tag{19}$$

Similarly, we sample the emission probability using the count for word-tag with $c(w_k, t_j) + \alpha_{WT}$ as the hyper-parameter.

3 Experiment Setup

Our experiment setup is similar to the ones used in (Gao & Johnson, 2007). They are summarized in Table 1:

Parameters	**Values**
Data Size	24k, 120k, 500k
Algorithm	EM, VB, GS(c,w), GS(c,s), GS(e,s), GS(e,w)
# of states	Chinese: 33 English: 50
α_{TT}	0.0001, 0.1, 0.5, 1
α_{WT}	0.0001, 0.1, 0.5, 1

Table 1: The list of experiments conducted. For the hyper-parameters $(\alpha_{TT}, \alpha_{WT})$, we try the combination of the adjacent pairs – (0.0001,0.0001), (0.1,0.0001), (0.0001,0.1), (0.1, 0.1), (0.1, 0.5), etc.

3.1 Data

For our experiments, we use the data set Chinese Penn Treebank (CTB) v5.0. The Chinese Treebank project began at the University of Pennsylvania in 1998 and the team created a set of annotation guidelines for word segmentation, POS tagging and bracketing (Xia, 2000; Xue et al., 2002; Xue et al., 2005). The version used in this paper is the Chinese Treebank 5.0 which consists of over 500k words and over 800k Chinese characters. The text comes from various sources including newswire, magazine articles, website news, transcripts from various broadcast news program.

Chinese POS tagging faces additional challenges because it has very little, if any, inflectional morphology. Words are not inflected with number, gender, case, or tense. For example, a word such as 毁灭 in Chinese corresponds to d*estroy /destroys /destroyed/destruction* in English. This fuels the discussion in Chinese NLP communities on whether the POS tags should be based on meaning or on syntactic distribution (Xia, 2000). If only the meaning is used, 毁灭 should be a verb all the time. If syntactic distribution is used, the word is a verb or a noun depending on the context. For the CTB, syntactic distribution is used, which complies with the principles of contemporary linguistics theories.

Following the experiment done for English in (Gao & Johnson, 2008), we split the data into three sizes: 24k words, 120k words and all words (500k), and used the same data set for training and testing. The idea is to track the effectiveness of an algorithm across different corpus sizes. Instead of using two different tag set sizes (17 and 50) as it is done for English POS tagging, we opt to keep the original 33 tag set for Chinese without further modification. In addition to reporting the results for English from (Gao & Johnson, 2008), we run additional experiments on English using only 500k words for comparison.

3.2 Decoding

For decoding, we use max marginal likelihood estimator (as opposed to using Viterbi algorithm) to assign a tag for each word in the result tag. (Gao & Johnson, 2008) finds that max marginal decoder performs as well as Viterbi algorithm and runs significantly faster as we can reuse the forward and backwards probabilities already calculated during the estimation and update step.

3.3 Hyperparameters

For the Bayesian approaches (VB and GS), we have a choice of hyperparameters. We choose uniform hyperparameters α_{TT} and α_{WT} instead of choosing a specific hyper-parameter for each of the tag-tag and word-tag distribution. The values for the hyper-parameters are chosen such that we can see more clearly the interactions between the two values. For GS, we use the notation GS(c,s) to denote collapsed sentence-based approach, GS(e,s) for explicit sentence based, GS(c,w) for collapsed word-based and GS(e,w) for explicit word based.

3.4 Evaluation Metrics

We use POS tagging accuracy as our primary evaluation method. There are two commonly used methods to map the state sequences from the system output to POS tags. In both methods, we first create a matrix where each row corresponds to a hidden state, each column corresponds to a POS tag, and each cell (i,j) represents the number of times a word position in the test data comes from the hidden state t_i according to the system output and the position has tag t_j according to the gold standard. In greedy 1-to-1 mapping, we find the largest value in the table – suppose the value is for the cell (i,j). We map state i to tag j, and remove both row i and column j from the table. We repeat the process until all the rows have been removed. Greedy many-to-1 allow multiple hidden states to map to a single POS tag. That is, when the highest value in the table is found, only the corresponding row is removed. In other words, we simply map each hidden state to the POS tag that the hidden state co-occurs with the most.

4 Results and Analysis

We compare and analyze the results between the different algorithms and between Chinese and English using Greedy 1-to-1 accuracy, Greedy many-to-1 accuracy.

4.1 Greedy 1-to-1 accuracy

When measure using 1-to-1 mapping, the best algorithm – Collapsed word based Gibbs Sampling GS(c,w) - achieve 0.358 in Chinese on the full data set but remains close to 0.499 in English for the full dataset. GS(c,w) outperforms other algorithm in almost all categories. But EM posts the highest relative improvement with an increase of 70% when the data size increases from 24k to 500k words. The full result is listed in Table 2.

	Greedy 1-to-1			
		24k	120k	500k
Chinese	**EM**	0.1483	0.1838	0.2406
	VB	0.1925	0.2498	0.3105
	GS(e,w)	0.2167	0.3108	0.3475
	GS(e,s)	0.2262	0.2596	0.3572
	GS(c,s)	0.2351	0.2931	**0.3577**
	GS(c,w)	**0.2932**	**0.3289**	0.3558
Eng	**EM**	0.1862	0.2930	0.3837
	VB	0.2382	0.3468	0.4327
	GS(c,w)	**0.3918**	**0.4276**	**0.4348**

Table 2: Tagging accuracy for Chinese and English with greedy 1-to-1 mapping. The English 24k and 120k results are taken from (Gao & Johnson 2008) with the 50-tag set.

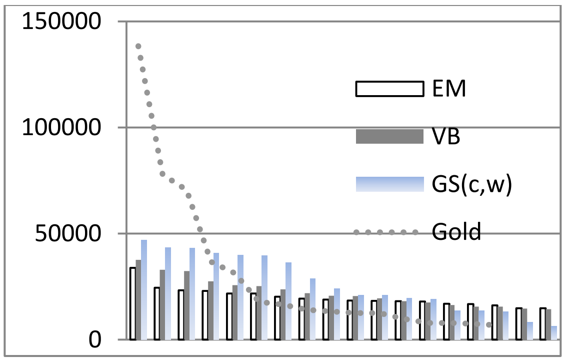

Figure 2: Tag distribution for 1-to-1 greedy mapping in Chinese 500k. Only the top 18 tags are shown. The figure compares the tag distribution between the gold standard for Chinese (33 tags) and the algorithm's results. The gold tags are shown as lines, and each algorithm's result is shown as bar graphs.

As expected, the increase in data size improves the accuracy as EM algorithm optimizes the likelihood better with more data. We ran additional experiments on English using a reduced 500k dataset to match the dataset used for Chinese; EM in this setting achieve an accuracy of 0.384 on average for 50 tags (down from 0.405). So even in the reduced data size setting, EM on English performs better than Chinese although the difference is reduced. We analyze the tag distribution of the 1-to-1 mapping. (Johnson, 2007) finds that EM generally assigns roughly as equal number of words for each state. In Figure 2, we find the same phenomenon for Chinese.

One of the advantages of Bayesian approaches (VB and GS) is that we can assign a prior to attempt to encourage a sparse model distribution. Despite using small values 0.0001 as hyperparameters, we find that the resulting distribution for number of words mapping to a particular state is very different from the gold standard.

4.2 Greedy many-to-1 accuracy

Collapsed Word Based Gibbs Sampler GS(c,w) is the clear winner for both English and Chinese unsupervised POS tagging. Table 3 shows the result of Greedy many-to-1 mapping for Chinese in different data size as well as English with the full data set. In Greedy many-to-1 mapping, GS(c,w) in both Chinese and English achieve 60%+ accuracy. In addition, the size of the dataset does not affect GS(c,w) as much as the other algorithms. In fact, the change from 24k to 500k dataset only increases the relative accuracy by less than 6%.

		Greedy many-to-1		
		24k	120k	500k
Chinese	**EM**	0.4049	0.4564	0.4791
	VB	0.4411	0.5023	0.5390
	GS(e,w)	0.4758	0.4969	0.5499
	GS(e,s)	0.4904	0.5369	0.5658
	GS(c,s)	0.5070	0.5701	0.5757
	GS(c,w)	**0.5874**	**0.6180**	**0.6213**
Eng	**EM**	0.2828	0.44135	0.5872
	VB	0.3595	0.48427	0.6025
	GS(c,w)	**0.5815**	**0.6529**	**0.6644**

Table 3: Many-to-1 accuracy for Chinese and English. The English 24k and 120k results are taken from (Gao & Johnson 2008) with the 50-tag set.

However, despite the relatively high accuracy, when analyzing the result, we notice that there are overwhelmingly many states which maps to a single POS tag (NN). Figure 3 shows the number of states mapping to different POS tags in Chinese over the 500k data size. There are a large number of states mapping to relatively few POS tags. In the most extreme example, for the POS tag NN, GS(e,s) assigns 18 (the most) hidden states, accounts for 44% of the word tokens mapping to NN whereas GS(e,w) assigns 13 states, which is actually the least among all the algorithms and accounts for 31% of the word tokens mapping to NN. Notice that we have only a total of 33 hidden states in our model. This means that over half the states are mapped to NN, which is a rather disappointing result. The actual empirical result for the gold standard in CTB is that only 27% of the word should be mapped to NN. For EM in particular, we see 17 states accounting for 42% of the words tagged as NN.

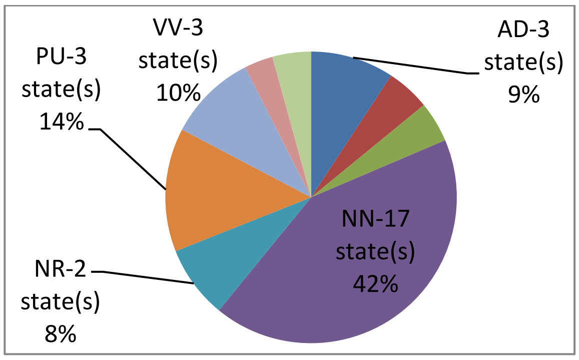

Figure 3: The distribution of POS tags based on the output EM algorithm in Chinese using the 500k dataset. Tag T-N-y% means that there are N hidden states mapped to the specific POS tag T accounting for y% of word tokens tagged with these N states by the EM algorithm.

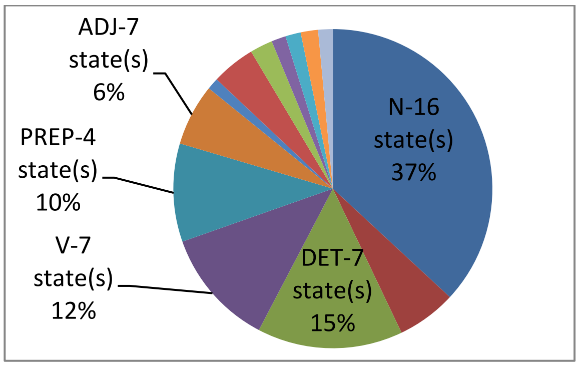

Figure 4: English tag distribution for EM using 500k dataset with 50 states mapping to the 17 pos tag set. Tag T-N-y% means that there are N hidden states mapped to the specific POS tag T accounting for y% of word tokens tagged with these N states.

We also ran additional experiments on the algorithms for English using a reduced data size of 500k to match that of our Chinese experiment to see whether we see the same phenomena. We notice that the tag distribution for English EM is more consistent to the empirical distribution found in the gold standard.

With the English 50 tag set with 500k words, we experiment with mapping the English 50 tag set result to the 17 tag set, we see that in Figure 4, 16 (of 50) states mapped to the N tag, accounting for 37% of the words in the dataset. This is close to the actual empirical distribution for English for 17 tags where N accounts for about 32%.

4.3 Convergence

We analyze how each algorithm converges to its local maxima. Figure 5 shows the change in greedy 1-to-1 accuracy over the 50% of the run.

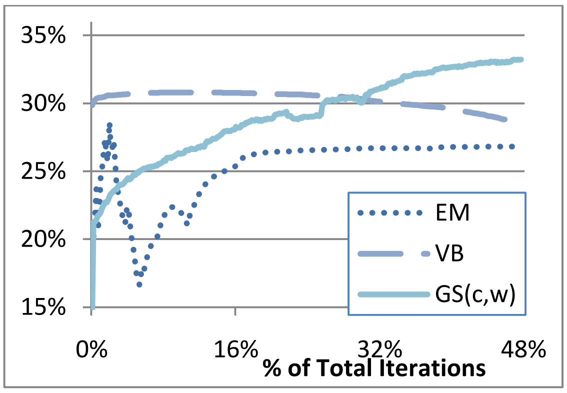

Figure 5: Greedy 1-to-1 accuracy of EM, VB and GS(c,w) over the first 50% of the algorithms' iterations for the Chinese 500k dataset. Note: the percentage of iterations is used here because each algorithms converge at a different number of iterations, thus the progress is scaled accordingly.

The greedy 1-to-1 accuracy actually fluctuates through the run. VB has an interesting dip at around 80% of its iteration before climbing to its max (not showing in the graph). All the Gibbs sampling variations follow a relatively steady hill climb before converging (only GS(c,w) is shown in Figure 5). EM is particularly interesting; Looking at the initial 15% of the algorithm's run, we can see that EM climbs to a “local” max very quickly before dropping and then slowly improving in its accuracy. The greedy 1-to-1 accuracy in the initial top is actually higher than the final convergence value in most runs. This initial peak in value following by a drop and then a slow hill climb in EM for Chinese POS tagging is consistent with the finding in (Johnson, 2007) for English POS tagging.

5 Conclusion and Future Work

We have only scratched the surface of the research in unsupervised techniques in Chinese NLP. We have established a baseline of EM, VB and GS against the CTB 5.0. The experiment shows that for both Chinese and English, GS(c,w) produces the best result. We have also found that Chinese performs rather poorly in the 1-to-1 accuracy when comparing against English in the same data size. We find that in many-to-1 mapping, we have a disproportionate large number of states mapping to individual POS tags comparing to the gold distribution and also in comparison to English against its gold distribution.

Graça et al. (2009) addresses the problem we observe in our resulting tag distributions in our model where EM, VB and GS fails to capture the shape of the true distribution. They propose a Posterior Regularization framework where it poses linear constraints on the posterior expectation. They define a set distributions Q over hidden states with a constraint on the expectation over the features. The log likelihood is penalized using the KL-divergence between the Q distribution and the model. The distributions that their model predicted are far more similar to the gold standard than traditional EM.

Liang and Klein (2009) propose some interesting error analysis techniques for unsupervised POS tagging. One of their analyses on EM is done by observing the approximation errors being created during each iteration of the algorithm's execution. We can also perform these analyses on VB and GS and observe the changes of output tags by starting from the Gold Standard distribution in EM and VB, and gold standard tags in GS. We can then follow how and which set of tags start to deviate from the gold standard. This will allow us to see which categories of errors (ex. noun-verb, adj-adv errors) occur most in these algorithms and how the error progresses.

Acknowledgment: This work is partly supported by the National Science Foundation Grant BCS-0748919. We would also like to thank three anonymous reviewers for their valuable comments.

References

Banko, M., & Moore, R. C. 2004. Part of Speech Tagging in Context. *In Proc. of the 20th International Conference on Computational Linguistics (COLING),* pp 556-561.

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

Besag, J. 2004. An introduction to Markov Chain Monte Carlo methods. *Mathematical Foundations of Speech and Language Processing*, pages 247–270. Springer, New York.

Chang, C.-H., & Chen, C.-D. 1993. HMM-Based Part-Of-Speech Tagging For Chinese Corpora. *Workshop On Very Large Corpora: Academic And Industrial Perspectives.*

Elworthy, D. 1994. Does Baum-Welch Re-estimation Help Taggers? In Proc. of *Applied Natural Language Processing Conference (ANLP),* pp 53-58.

Gao, J., & Johnson, M. 2008. A comparison of Bayesian estimators for unsupervised Hidden Markov Model POS taggers. *In Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP),* pp 344-352.

Geman, S., & Geman, D. 1984. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp 721–741.

Goldwater, S., & Griffiths, T. 2007. A Fully Bayesian Approach to Unsupervised Part-of-Speech Tagging. *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL)* , pp 744-751.

Graca, M.J., Ganchev, K., Taskar B. & Pereira, F. 2009. Posterior vs. Parameter Sparsity in Latent Variable. *Advances in Neural Information Processing Systems 22 (NIPS)*. MIT Press.

Haghighi, A., & Klein, D. 2006. Prototype-driven learning for sequence models. *In Proceedings of the Human Language Technology Conference (HLT- NAACL)* , pp 320-327.

Huang, Z., Eidelman, V., Harper, M. 2009. Improving A Simple Bigram HMM Part-of-Speech Tagger by Latent Annotation and Self-Training. In Proc. of *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), Companion Volume: Short Papers.*

Johnson, M. 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),* pp 296-305.

Liang, P., & Klein, D. 2008. Analyzing the errors of unsupervised learning. *The Forty Sixth Annual Meeting of the Association for Computational Linguistics (ACL),* pp 879–887. Columbus, OH.

MacKay, D. J., & Peto, L. C. 1994. A Hierarchical Dirichlet Language Model. *Natural Language Engineering*, 1-19.

Manning, C. & Schutze, H. 1999. *Foundations of Statistical Natural Language Processing.* The MIT Press, Cambridge, Massachusetts.

Merialdo, B. 1994. Tagging English text with a probabilistic model. *Computational Linguistics,* 20(2).

Ng, H. T., & Low, J. K. 2004. Chinese Part-Of-Speech Tagging: One-At-A-Time Or All-At-Once? Word-Based Or Character-Based? In Proc. of *EMNLP*.

Thorsten Brants, 2000. TnT - A Statistical Part-of-Speech Tagger. In *Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP)*, Seattle, WA.

Toutanova, K., & Johnson, M. 2007. A Bayesian LDA-based model for semi-supervised. In *Proceedings of NIPS 21* .

Wolfram Mathematica. (2009, 10 3). *Random Number Generation*. http://reference.wolfram.com/mathematica/tutorial/RandomNumberGeneration.html .

Xia, F. 2000. The Part-of-Speech Guidelines for the Penn Chinese Treebank (3.0). University of *Pennsylvania: IRCS Report 00-07.*

Xue, N., Chiou, F.-D., & Palmer, M. 2002. Building a Large-Scale Annotated Chinese Corpus. *Proceedings of the 19th. International Conference on Computational Linguistics (COLING).* Taipei, Taiwan.

Xue, N., Xia, F., Chiou, F.-D., & Palmer, M. 2005. The Penn Chinese TreeBank: Phrase Structure Annotation of a Large Corpus. *Natural Language Engineering*, 11(2), pp 207-238.