Efficient Methods for Topic Model Inference on Streaming Document Collections

Limin Yao, David Mimno, and Andrew McCallum
Department of Computer Science
University of Massachusetts, Amherst
{lmyao, mimno, mccallum}@cs.umass.edu

ABSTRACT

Topic models provide a powerful tool for analyzing large text collections by representing high dimensional data in a low dimensional subspace. Fitting a topic model given a set of training documents requires approximate inference techniques that are computationally expensive. With today's large-scale, constantly expanding document collections, it is useful to be able to infer topic distributions for new documents without retraining the model. In this paper, we empirically evaluate the performance of several methods for topic inference in previously unseen documents, including methods based on Gibbs sampling, variational inference, and a new method inspired by text classification. The classification-based inference method produces results similar to iterative inference methods, but requires only a single matrix multiplication. In addition to these inference methods, we present SparseLDA, an algorithm and data structure for evaluating Gibbs sampling distributions. Empirical results indicate that SparseLDA can be approximately 20 times faster than traditional LDA and provide twice the speedup of previously published fast sampling methods, while also using substantially less memory.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Experimentation, Performance, Design

Keywords

Topic modeling, inference

1. INTRODUCTION

Statistical topic modeling has emerged as a popular method for analyzing large sets of categorical data in applications from text mining to image analysis to bioinformatics. Topic models such as latent Dirichlet allocation (LDA) [3] have the ability to identify interpretable low dimensional components in very high dimensional data. Representing documents as topic distributions rather than bags of words reduces the effect of lexical variability while retaining the overall semantic structure of the corpus.

Although there have recently been advances in fast inference for topic models, it remains computationally expensive. Full topic model inference remains infeasible in two common situations. First, data streams such as blog posts and news articles are continually updated, and often require real-time responses in computationally limited settings such as mobile devices. In this case, although it may periodically be possible to retrain a model on a snapshot of the entire collection using an expensive "offline" computation, it is necessary to be able to project new documents into a latent topic space rapidly. Second, large scale collections such as information retrieval corpora and digital libraries may be too big to process efficiently. In this case, it would be useful to train a model on a random sample of documents, and then project the remaining documents into the latent topic space independently using a MapReduce-style process. In both cases there is a need for accurate, efficient methods to infer topic distributions for documents outside the training corpus. We refer to this task as "inference", as distinct from "fitting" topic model parameters from training data.

This paper has two main contributions. First, we present a new method for topic model inference in unseen documents that is inspired by techniques from discriminative text classification. We evaluate the performance of this method and several other methods for topic model inference in terms of speed and accuracy relative to fully retraining a model. We carried out experiments on two datasets, NIPS and Pubmed. In contrast to Banerjee and Basu [1], who evaluate different statistical models on streaming text data, we focus on a single model (LDA) and compare different inference methods based on this model. Second, since many of the methods we discuss rely on Gibbs sampling to infer topic distributions, we also present a simple method, SparseLDA, for efficient Gibbs sampling in topic models along with a data structure that results in very fast sampling performance with a small memory footprint. SparseLDA is approximately 20 times faster than highly optimized traditional LDA and twice the speedup of previously published fast sampling methods [7].

2. BACKGROUND

A statistical topic model represents the words in documents in a collection $\mathcal{W}$ as mixtures of T"topics," which

KDD'09, June 28–July 1, 2009, Paris, France.

are multinomials over a vocabulary of size V. Each document d is associated with a multinomial over topics θ_d. The probability of a word type w given topic t is represented by $\phi_{w|t}$. We refer to the complete VxT matrix of topic-word probabilities as Φ. The multinomial parameters θ_d and ϕ_t are drawn from Dirichlet priors with parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ respectively. In practice we use a symmetric Dirichlet with $\beta = .01$ for all word types and a T-dimensional vector of distinct positive real numbers for α. Each token w_i in a given document is drawn from the multinomial for the topic represented by a discrete hidden indicator variable z_i.

Fitting a topic model given a training collection $\mathcal{W}$ involves estimating both the document-topic distributions, θ_d, and the topic-word distributions, Φ. MAP estimation in this model is intractable due to the interaction between these terms, but relatively efficient MCMC and variational methods are widely used [4, 3]. Both classes of methods can produce estimates of Φ, which we refer to as $\hat{\Phi}$. In the case of collapsed Gibbs sampling, the Markov chain state consists of topic assignments $\boldsymbol{z}$ for each token in the training corpus. An estimate of $P(w|t)$ can be obtained from the predictive distribution of a Dirichlet-multinomial distribution:

$$\hat{\phi}_{w|t} = \frac{\beta + n_{w|t}}{\beta V + n_{\cdot|t}} \quad (1)$$

where $n_{w|t}$ is the number of tokens of type w assigned to topic t and $n_{\cdot|t} = \sum_w n_{w|t}$.

The task of topic model inference on unseen documents is to infer θ for a document d not included in $\mathcal{W}$. The likelihood function for θ is

$$\mathcal{L}(\theta|\boldsymbol{w}, \Phi, \boldsymbol{\alpha}) = \prod_i \phi_{w_i|z_i}\theta_{z_i} \times \frac{\Gamma(\sum_t \alpha_t)}{\prod_t \Gamma(\alpha_t)} \prod_t \theta_t^{\alpha_t - 1} \quad (2)$$

$$\propto \prod_i \phi_{w_i|z_i} \prod_t \theta_t^{n_{t|d} + \alpha_t - 1}, \quad (3)$$

where $n_{t|d}$ is the total number of tokens in the document assigned to topic t. Even with a fixed estimate $\hat{\Phi}$, MAP estimation of θ is intractable due to the large number of discrete hidden variables $\boldsymbol{z}$. Sections 3 through 5 present an array of approximate inference methods that estimate $\hat{\theta}$, which are empirically evaluated in the remaining sections of the paper.

3. SAMPLING-BASED INFERENCE

We evaluate three different sampling-based inference methods for LDA. Gibbs sampling is an MCMC method that involves iterating over a set of variables $z_1, z_2, ...z_n$, sampling each z_i from $P(z_i|\boldsymbol{z}_{\setminus i}, \boldsymbol{w})$. Each iteration over all variables is referred to as a Gibbs sweep. Given enough iterations, Gibbs sampling for LDA [4] produces samples from the posterior $P(\boldsymbol{z}|\boldsymbol{w})$. The difference between the three methods we explore is in the set of variables $\boldsymbol{z}$ that are sampled, as illustrated in Figure 1, and which portion of the complete data is used in estimating $\hat{\Phi}$.

3.1 Gibbs1: Jointly resample all topics

In this method we define the scope of a single Gibbs sweep to be the hidden topic variables for the entire collection, including both the original documents and the new documents. After sampling the topic variables for the training documents to convergence without the new documents, we randomly initialize topic variables for the new documents and continue sampling over all documents until the model converges again. We can then estimate the topic distribution θ_d for a given document given a single Markov chain state as

$$\hat{\theta}_{t|d} = \frac{\alpha_t + n_{t|d}}{\sum_{t'} \alpha_{t'} + n_{\cdot|d}}, \quad (4)$$

where $n_{\cdot|d}$ is the length of the document. Increasingly accurate estimates can be generated by averaging over values of Eq. 4 for multiple Markov chain states, but may cause problems due to label swapping.

Inference method Gibbs1 is equivalent to fitting a new model for the complete data, including both the original documents and the new documents, after separately initializing some of the topic variables using Gibbs sampling. This method is as computationally expensive as simply starting with random initializations for all variables. It is useful, however, in that we can use the same initial model for all other inference methods, thereby ensuring that the topics will roughly match up across methods. We consider this inference procedure the most accurate, and the topic distribution for each test document estimated using Gibbs1 as a reference for evaluating other inference methods.

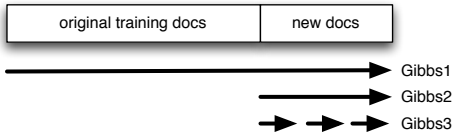

Figure 1: The three Gibbs sampling-based methods iterate over a set of words, updating the topic assignment for each word given the topic assignments for the remaining words. The methods vary only in the set of topic assignments they consider: Gibbs1 samples new topic assignments for the entire corpus, including the original training documents; Gibbs2 samples assignments for only the new documents, holding the parameters for the training corpus fixed; Gibbs3 samples each new document independently.

3.2 Gibbs2: Jointly resample topics for all new documents

Inference method Gibbs2 begins with the same initialization as Gibbs1, but saves computation by holding all of the topic assignments for the training documents fixed. Under this approximation, one Gibbs sweep only requires updating topic assignments for the new documents.

Inference involves sampling topic assignments for the training data as in Gibbs1, randomly assigning values to the topic indicator variables $\boldsymbol{z}$ for the new documents, and then sampling as before, updating $\hat{\Phi}$ as in Eq. 1 after each variable update.

3.3 Gibbs3: Independently resample topics for new documents

Inference method Gibbs2 performs Gibbs sampling as a batch. As such, it samples from the posterior distribution over all $\boldsymbol{z}$ variables in the new documents given all the words in the new documents, accessing all the new documents in each iteration. Unfortunately, handling topic inference in a

batch manner is both unrealistic in time-sensitive streaming document collections, and inefficient because it cannot be parallelized across documents without substantial interprocess communication.

Gibbs3 is an online version, which processes all documents independently. When a test document arrives, we sample topics for a number of iterations using only topic-word counts in $\hat{\Phi}$ from the training corpus and the current document. For the next incoming document we reset $\hat{\Phi}$ to include only counts from the training corpus and that new document.

This algorithm differs from the previous two methods in that it produces estimates of θ_d given only the words in the training documents and in document d. Gibbs1 and Gibbs2 produce estimates given the entire data set.

3.4 Time- and Memory-Efficient Gibbs Sampling for LDA

The efficiency of Gibbs sampling-based inference methods depends almost entirely on how fast we can evaluate the sampling distribution over topics for a given token. We therefore present SparseLDA, our new algorithm and data structure that substantially improves sampling performance. Although we apply this method to topic inference on new documents, the method is applicable to model fitting as well.

The probability of a topic z in document d given an observed word type w is

$$P(z = t|w) \quad \propto \quad (\alpha_t + n_{t|d})\frac{\beta + n_{w|t}}{\beta V + n_{\cdot|t}}. \tag{5}$$

Sampling from this distribution involves calculating the unnormalized weight in Eq. 5, which we refer to as $q(z)$, for each topic; sampling a random variable $U \sim \mathcal{U}(0, \sum_z q(z))$; and finding t such that $\sum_{z=1}^{t-1} q(z) < U < \sum_{z=1}^{t} q(z)$. This algorithm requires calculating $q(z)$ for all topics in order to determine the normalizing constant for the distribution $\sum_z q(z)$, even though probability mass is generally concentrated on a small number of topics. Porteous et al. [7] approach this problem by iteratively refining an approximation to $\sum_z q(z)$. We take an arguably simpler approach by caching most of the computation required to compute the normalizing constant. By rearranging terms in the numerator, we can divide Eq. 5 into three parts:

$$P(z = t|w) \propto \frac{\alpha_t\beta}{\beta V + n_{\cdot|t}} + \frac{n_{t|d}\beta}{\beta V + n_{\cdot|t}} + \frac{(\alpha_t + n_{t|d})n_{w|t}}{\beta V + n_{\cdot|t}}. \tag{6}$$

Note that the first term is constant for all documents and that the second term is independent of the current word type w. Furthermore, $\sum_z q(z)$ is equal to the sum over topics of each of the three terms in Equation 6:

$$s \quad = \quad \sum_t \frac{\alpha_t\beta}{\beta V + n_{\cdot|t}} \tag{7}$$

$$r \quad = \quad \sum_t \frac{n_{t|d}\beta}{\beta V + n_{\cdot|t}} \tag{8}$$

$$q \quad = \quad \sum_t \frac{(\alpha_t + n_{t|d})n_{w|t}}{\beta V + n_{\cdot|t}}. \tag{9}$$

This process divides the full sampling mass into three "buckets." We can now sample $U \sim \mathcal{U}(0, s + r + q)$. If $U < s$, we have hit the "smoothing only" bucket. In this case, we can step through each topic, calculating and adding up $\frac{\alpha_t\beta}{|V|\beta+n_t}$ for that topic, until we reach a value greater than x. If $s < x < (s + r)$, we have hit the "document topic" bucket. In this case, we need only iterate over the set of topics t such that $n_{t|d} \neq 0$ — a number that is usually substantially less than the total number of topics. Finally, if $x > (s + r)$, we have hit the "topic word" bucket, and we need only consider topics such that $n_{w|t} \neq 0$. Again, this number is usually very small compared to T.

The values of the three components of the normalization constant, s, r, q, can be efficiently calculated. The constant s only changes when we update the hyperparameters $\boldsymbol{\alpha}$. The constant r depends only on the document-topic counts, so we can calculate it once at the beginning of each document and then update it by subtracting and adding values for the terms involving the old and new topic at each Gibbs update. This process takes constant time, independent of the number of topics.

The topic word constant q changes with the value of w, so we cannot as easily recycle earlier computation. We can, however, substantially improve performance by observing that the expression for q can be broken into two components:

$$q \quad = \quad \sum_t \left[\frac{\alpha_t + n_{t|d}}{\beta V + n_{\cdot|t}} \times n_{w|t}\right]. \tag{10}$$

The coefficient $\frac{\alpha_t+n_{t|d}}{|V|\beta+n_t}$ can therefore be cached for every topic, so calculating q for a given w consists of one multiply operation for every topic such that $n_{w|t} \neq 0$. As $n_{t|d} = 0$ for almost all topics in any given document, this vector of coefficients will also almost entirely consist of only $\frac{\alpha_t}{|V|\beta+n_t}$, so we can save additional operations by caching these coefficients across documents, only updating those topics that have non-zero counts in the current document as we begin each document, and resetting those values to the $\boldsymbol{\alpha}$-only values as we complete sampling for each document.

If the values of α and β are small, q will take up most of the total mass. Empirically, we find that more than 90% of samples fall within this bucket. In a Dirichlet-multinomial distribution with small parameter magnitudes, the likelihood of the distribution is roughly proportional to the concentration of the counts on a small number of dimensions. We find that the wallclock time per iteration is roughly proportional to the likelihood of the model. As the sampler approaches a region of high probability, the time per iteration declines, leveling off as the sampler converges.

Clearly, the efficiency of this sampling algorithm depends on the ability to rapidly identify topics such that $n_{w|t} \neq 0$. Furthermore, as the terms in Eq. 10 are roughly proportional to $n_{w|t}$, and since we can stop evaluating terms as soon as the sum of terms exceeds $U - (s + r)$, it is desirable to be able to iterate over non-zero topics in descending order. We now present a novel data structure that meets these criteria.

We encode the tuple $(t, n_{w|t})$ in a single 32 bit integer by dividing the bits into a *count* segment and a *topic* segment. The number of bits in the topic segment is the smallest m such that $2^m \geq T$. We encode the values by shifting $n_{w|t}$ left by m bits and adding t. We can recover $n_{w|t}$ by shifting the encoded integer right m bits and t by a bitwise *and* with a "topic mask" consisting of m 1s. This encoding has two primary advantages over a simple implementation that stores $n_{w|t}$ in an array indexed by t for all topics. First, in natural languages most word types occur rarely. As the

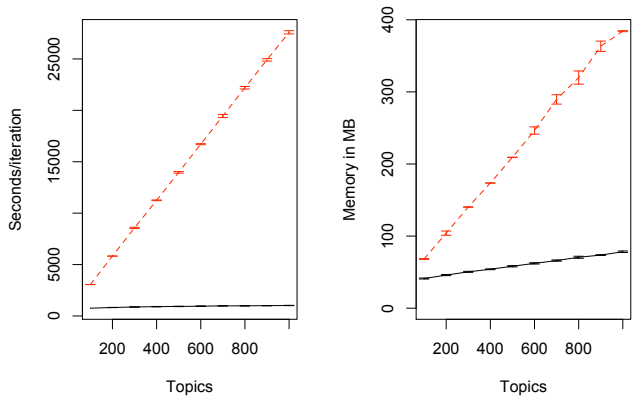

Figure 2: A comparison of time and space efficiency between standard Gibbs sampling (dashed red lines) and the SparseLDA algorithm and data structure presented in this paper (solid black lines). Error bars show the standard deviation over five runs.

encoding no longer relies on the array index, if a word type w only occurs three times in the corpus, we need only reserve an array of at most three integers for it rather than T. Second, since the count is in the high bits, we can sort the array using standard sorting implementations, which do not need to know anything about the encoding.

By storing encoded values of $(t, n_{w|t})$ in reverse-sorted arrays, we can rapidly calculate q, sample U, and then (if $U > (s+r)$) usually within one calculation find the sampled t. Maintaining the data structure involves reencoding values for updated topics and ensuring that the encoded values remain sorted. As $n_{w|t}$ changes by at most one after every Gibbs update, a simple bubble sort is sufficient.

In order to evaluate the efficiency of SparseLDA, we measured the average time per iteration and memory usage,[1] shown in Figure 2 averaged over five runs for each value of T on NIPS data. As we increase the number of topics, the time per iteration increases slowly, with much lower overall time than a carefully optimized implementation of standard Gibbs sampling. Memory usage is also substantially lower and grows more slowly than standard Gibbs sampling. Empirical comparisons with the FastLDA method of Porteous et al. [7] are difficult, but on the standard NIPS corpus with 800 topics, we found the ratio of per-iteration times between standard Gibbs sampling and SparseLDA to be approximately 20, compared to a speedup of approximately 11 times reported for FastLDA over standard Gibbs sampling using the same corpus and number of topics with $\alpha_t = 0.001$. In addition, FastLDA does not address memory efficiency, one of the primary benefits of SparseLDA.

4. VARIATIONAL INFERENCE

Another class of approximate inference method widely used in fitting topic models is variational EM. Variational inference involves defining a parametric family of distributions that forms a tractable approximation to an intractable true joint distribution. In the case of LDA, Blei, Ng, and Jordan [3] suggest a factored distribution consisting of a variational Dirichlet distribution $\tilde{\alpha}_d$ for each document and a variational multinomial $\tilde{\gamma}_{di}$ over topics for each word position in the document. The parameters of these distributions can then be iteratively fitted using the following update rules:

$$\tilde{\alpha}_{dt} = \alpha_t + \sum_i \tilde{\gamma}_{dit} \quad (11)$$

$$\tilde{\gamma}_{dit} \propto \hat{\phi}_{w_i|t} \exp(\Psi(\tilde{\alpha}_{dt})), \quad (12)$$

where $\Psi(x)$ is the digamma function. Variational approximations converge in fewer iterations than Gibbs sampling, but each iteration generally takes substantially longer due to the calculation of exp and Ψ functions.

We evaluate a variational method based on the update rules specified. Given a new document, we initialize $\tilde{\boldsymbol{\alpha}}_d$ to $\boldsymbol{\alpha}$ and clamp $\hat{\Phi}$ to the expression in Eq. 1 evaluated using the topic-word counts from the training documents. We then apply the update rules in Eq. 11 and 12 in turn until convergence. We then estimate $\hat{\theta}_t \propto \tilde{\alpha}_t$. This method is a variational equivalent to inference method Gibbs3, being appropriate for parallelized, streaming environments. More complicated variational distributions have been shown to have lower bias [9], but are generally more computationally intensive per iteration than the simple fully factored variational distribution.

5. CLASSIFICATION-BASED INFERENCE

The previous inference methods theoretically require many iterations to allow a Markov chain to achieve its stationary distribution or to fit a variational approximation. In some settings, however, it is necessary to estimate a topic distribution as fast as possible. As an alternative to iterative methods, we propose a classification-based approach to predicting θ for a new document.

In this task, each document has a labeled topic distribution, obtained from the trained topic model. Specifically, each document can be classified as multiple classes, each with a probability. This setting is an extension to the traditional classification task, in which an instance has only one class membership. In a traditional classifier, the training data is represented as $D = \{(\mathbf{x}_i, \mathbf{y}_i) \mid i = 1 \ldots n\}$, where $\mathbf{x}_i$ is a representation of the document (e.g. a feature vector) and $\mathbf{y}_i$ is $\mathbf{x}_i$'s label. $\mathbf{y}_i$ is typically a vector of indicator variables, such that if $\mathbf{x}_i$ belongs to the jth class, only the jth element of $\mathbf{y}_i$ will be 1, and other elements will be 0. For simplicity, we often use the class label instead of the indicator vector to represent y_i. For our task, we allow each element of $\mathbf{y}_i$ to take values between 0 and 1 and the sum of all elements should be 1, so that $\mathbf{y}_i$ indicates a probabilistic distribution over all classes. Classes in this setting are equivalent to topics. We investigate two classifiers in this paper, one of which is useful only as a baseline.

5.1 Maximum Entropy Classifier (MaxEnt)

We first extend a traditional MaxEnt or logistic regression classifier to our new task. In the simple scenario in which each instance has one fully observed class label, any real-valued function $f_i(\mathbf{x}, y)$ of the object $\mathbf{x}$ and class y can be treated as a *feature*. The constraints are the expected values of features computed using training data. Given a set $\mathcal{Y}$

[1]Memory use in Java was calculated as `runtime.totalMemory() - runtime.freeMemory()`

of classes, and features f_k, the parameters to be estimated λ_k, the learned distribution $p(y|\mathbf{x})$ is of the parametric exponential form [2]:

$$P(y|\mathbf{x}) = \frac{\exp\left(\sum_k \lambda_k f_k(\mathbf{x}, y)\right)}{\sum_y \exp\left(\sum_k \lambda_k f_k(\mathbf{x}, y)\right)}. \quad (13)$$

Given training data $D = \{\langle \mathbf{x_1}, y_1\rangle, \langle \mathbf{x_2}, y_2\rangle, ..., \langle \mathbf{x_n}, y_n\rangle\}$, the log likelihood of parameters Λ is

$$l(\Lambda|D) = \log\left(\prod_{i=1}^{n} p(y_i|\mathbf{x_i})\right) - \sum_k \frac{\lambda_k^2}{2\sigma^2}$$
$$= \sum_{i=1}^{n}\sum_k \left(\lambda_k f_k(\mathbf{x_i}, y_i) - \log Z_\Lambda(\mathbf{x})\right) - \sum_k \frac{\lambda_k^2}{2\sigma^2}, \quad (14)$$

The last term represents a zero-mean Gaussian prior on the parameters, which reduces overfitting and provides identifiability. We find values of Λ that maximize $l(\Lambda|D)$ using a standard numerical optimizer. The gradient with respect to feature index k is

$$\frac{\delta(\Lambda|D)}{\delta\lambda_k} = \sum_{i=1}^{n}\left(f_k(\mathbf{x_i}, y_i) - \sum_y f_k(\mathbf{x_i}, y)p(y|\mathbf{x_i})\right) - \frac{\lambda_k}{\sigma^2}. \quad (15)$$

In the topic distribution labeling task, each data point has a topic distribution, and is represented as $(\mathbf{x_i}, \mathbf{y_i})$. We can also use the maximum log likelihood method to solve this model. The only required change is to substitute $\mathbf{y}$ for y. Using a distribution changes the empirical features of the data ($f_k(\mathbf{x_i}, y_i)$), also known as the constraints in a maximum entropy model, which are used to compute the gradient. Whereas in a traditional classifier we use $f_k(\mathbf{x_i}, y_i)$ as empirical features, we now use $f_k(\mathbf{x_i}, \mathbf{y_i})$ instead, where $\mathbf{y_i}$ is the labeled topic distribution of the ith data point. Suppose that we have two classes (i.e. topics) and each instance can contain two features (i.e. words). Training data might consist of $\mathbf{x} = (x_1, x_2), y = 1$ for a traditional classifier and $\mathbf{x} = (x_1, x_2), \mathbf{y} = (p_1, p_2)$ for a topic distribution classifier, such that p_1 and p_2 are the proportions of topic 1 and topic 2 for data point $\mathbf{x}$ and $p_1 + p_2 = 1$. Empirical features (sufficient statistics of training data) for traditional classifier would be $(x_1, 1)$ and $(x_2, 1)$. While the empirical features for a topic distribution classifier would be $(x_1, 1)$, $(x_2, 1)$, $(x_1, 2)$, and $(x_2, 2)$, with the first two weighted by p_1, and the remaining two weighted by p_2. This substitution changes the penalized log likelihood function:

$$l(\Lambda \mid D) = \log\left(\prod_{i=1}^{n} p(\mathbf{y_i}|\mathbf{x_i})\right) - \sum_k \frac{\lambda_k^2}{2\sigma^2}$$
$$= \sum_{i=1}^{n}\sum_k \left(\lambda_k f_k(\mathbf{x_i}, \mathbf{y_i}) - \log Z_\Lambda(\mathbf{x})\right) - \sum_k \frac{\lambda_k^2}{2\sigma^2}, \quad (16)$$

Correspondingly, the gradient at feature index k is:

$$\frac{\delta(\Lambda|D)}{\delta\lambda_k} = \sum_{i=1}^{n}\left(\sum_y p_i(y) f_k(\mathbf{x_i}, y) - \sum_y f_k(\mathbf{x_i}, y)p(y|\mathbf{x_i})\right) - \frac{\lambda_k}{\sigma^2}. \quad (17)$$

Where $p_i(y)$ stands for the probability of topic y in the current instance, i.e. one of the elements of $\mathbf{y_i}$.

Once we have trained a topic proportion classifier, we can use it to estimate θ for a new document. We compute the scores for each topic using Eq. 13. This process is essentially a table lookup for each word type, so generating $\hat{\theta}$ requires a single pass through the document.

In experiments, we found that the output of the topic proportion classifier is often overly concentrated on the single largest topic. We therefore introduce a temperature parameter τ. Each feature value is weighted by $\frac{1}{\tau}$. Values of $\tau < 1$ increase the peakiness of the classifier, while values $\tau > 1$ decrease peakiness. We chose 1.2 for NIPS data and 0.9 for Pubmed data based on observation of the peakiness of predicted $\hat{\theta}$ values for each corpus.

5.2 Naive Bayes Classifier

From the trained topic model, we can estimate Φ, a matrix with T (#topics) rows and W (#words) columns representing the probability of words given topics. Combined with a uniform prior over topic distributions, we can use this matrix as a classifier, similar to the classifier we obtained from MaxEnt. This method performs poorly, and is presented only as a baseline in our experiments. A document d is represented as a vector, with each element an entry in the vocabulary, denoted as w and the value as the number of times that word occurs in the document, denoted as $n_{w|d}$. Using Bayes' rule, the score for each topic is:

$$\text{Score}(z = t) = \sum_w \hat{\phi}_{w|t} n_{w|d} \quad (18)$$

The estimated θ distribution is then simply the normalized scores.

In experiments, we compare both classification methods against the inference methods discussed in previous sections. The two classifiers take less time to predict topic distributions, as they do not require iterative updates. Provided they can achieve almost the same accuracy as the three inference methods or their performance is not much worse, for some particular task which requires real-time response, we can choose classification-based inference methods instead of sampling based or variational updated methods. The choice of estimator can be a trade-off between accuracy and time efficiency.

5.3 Hybrid classification/sampling inference

A hybrid classification/sampling-based approach can be constructed by generating an estimate of θ_d given $\boldsymbol{w}_d$ using the MaxEnt classifier and then repeatedly sampling topic indicators $\boldsymbol{z}$ given $\hat{\theta}$ and $\hat{\Phi}$. Note that given $\hat{\theta}$, $P(z_i|w_i) \propto \hat{\theta}_t \hat{\phi}_{w|t}$ is independent of all $\boldsymbol{z}_{\setminus i}$. After the initial cost of setting up sampling distributions, sampling topic indicators for each word can be performed in parallel and at minimal cost. After collecting sampled $\boldsymbol{z}$ indicators, we can re-estimate the topic distribution $\hat{\theta}$ according to the topic assignments as in Eq. 4. In our experiments, we find that this re-sampling process results in more accurate topic distributions than MaxEnt alone.

6. EMPIRICAL RESULTS

In this section we empirically compare the relative accuracy of each inference method. We train a topic model on training data with a burn-in period of 1000 iterations for all inference methods. We explore the sensitivity of each method to the number of topics, the proportion between

Table 1: Top five topics predicted by different methods for a testing document

Method	θ_t	Highest probability words in topic
Gibbs1	0.4395	learning generalization error
	0.2191	function case equation
	0.0734	figure time shown
	0.0629	information systems processing
	0.0483	training set data
MaxEnt	0.2736	learning generalization error
	0.2235	function case equation
	0.0962	figure time shown
	0.0763	information systems processing
	0.0562	learning error gradient

training documents and "new" documents, and the effect of topic drift in new documents.

We evaluate different inference methods using two data sets. The first is 13 years of full papers published in the NIPS conference, in total 1,740 documents. The second is a set of 51,616 journal article abstracts from Pubmed. The NIPS data set contains fewer documents, but each document is longer. NIPS has around 70K unique words and 2.5M tokens. Pubmed has around 105.4K unique words and about 7M tokens. We also carried out experiments on New York Times data from LDC. We used the first six months of 2007, comprising 39,218 documents, around 12M tokens, about 900K unique words. We preprocessed the data by removing stop words.

We implemented the three sampling-based inference methods (using SparseLDA), the variational updated method, and the classification-based methods in the MALLET toolkit [5]. They will be available as part of its standard open-source release.

6.1 Evaluation Measures

It is difficult to evaluate topic distribution prediction results, because the "true" topic distribution is unobservable. We can, however, compare different methods with each other. We consider the Gibbs1 inference method to be the most accurate, as it is closest to Gibbs sampling over the entire corpus jointly, a process that is guaranteed to produce samples from the true posterior over topic distributions. In order to determine whether sampling to convergence is necessary, for inference methods Gibbs2 and Gibbs3, we report results using 1000 iterations of sampling (Gibbs2 and Gibbs3) and two iterations (Gibbs2.S and Gibbs3.S), which is the minimum number of Gibbs sweeps for all topic indicators to be sampled using information from all other tokens.

We represent the prediction results of each method as a T-dimensional vector $\hat{\theta}_d$ for each document. We compare methods using three metrics. In all results we report the average of these measures over all "new" documents.

1. **Cosine distance** This metric measures the angle between two vectors P and Q representing $\hat{\theta}_d$ as estimated by two different inference methods:

$$D_{cos}(P\|Q) = \frac{\sum_t p_t q_t}{\|P\|\|Q\|} \tag{19}$$

   Values closer to 1.0 represent closely matched distributions.

2. **KL Divergence** Another metric between distributions P and Q is KL divergence:

$$D_{KL}(P\|Q) = \sum_t p_t \log \frac{p_t}{q_t}. \tag{20}$$

   Smaller values of this metric represent closer distributions. We use the "gold standard" inference method as P.

3. **F1** The previous two metrics measure the divergence between probability distributions. As shown in Table 1, however, it is common for estimators to produce rather different distributions while maintaining roughly the same overall ordering of topics. In this metric, we attempt to predict the set of topics that account for the largest probability mass. Specifically, we sort the entries in $\hat{\theta}_d$ for a given inference method in descending order and select a set of topics $\mathcal{T}$ such that $\sum_{t\in\mathcal{T}} \hat{\theta}_{td} \leq 0.8$. We can then treat $\mathcal{T}_{Gibbs1}$ as the correct topics and measure the precision and recall of $\mathcal{T}_m$ for all other methods m. The F1 measure is the harmonic mean between the precision and recall. Note that F1 does not take into account the order of topics, only whether they are in the set of topics selected by the gold standard method. Values close to 1.0 represent better matches.

For classification based inference methods we use unigram counts as input features. Normalized term frequency features (term counts normalized by document length) produced poorer results. We also tried including word-pair features, on the intuition that the power of topic models comes from cooccurrence patterns in words, but these features greatly increased inference time, never improved results over unigram features, and occasionally hurt performance. We hypothesize that the power of unigram features in the discriminatively trained MaxEnt classifier may be a result of the fact that the classifier can assign *negative* weights to words as well as positive weights. This capability provides extra power over the Naïve Bayes classifier, which cannot distinguish between words that are strongly negatively indicative of a topic and words that are completely irrelevant.

6.2 Discussion

We first compare each method to the Gibbs1 inference method, which is equivalent to completely retraining a model given the original data and new data. We split the NIPS data set into training and testing documents in a 7:3 ratio and run an initial model on the training documents with 70 topics. We explore the effect of these settings later in this section.

Figure 3 shows results for the three evaluation metrics. The converged sampling methods Gibbs2 and Gibbs3 are closest to Gibbs1 in terms of cosine distance, F1, and KL divergence, but do not exactly match. The two-iteration versions Gibbs2.S and Gibbs3.S are close to Gibbs1 in terms of cosine distance and KL divergence, but MaxEnt and Variational EM are closer in terms of F1. Hybrid MaxEnt consistently outperforms MaxEnt. Figure 4 shows similar measures vs. Gibbs2, arguably a more meaningful comparison

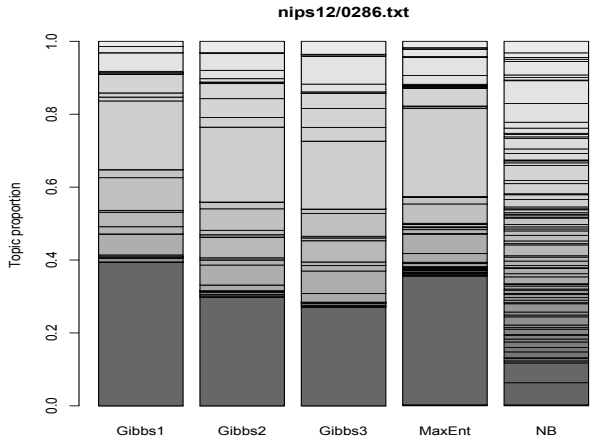

Figure 5: Examples of topic distribution obtained by different methods (NIPS)

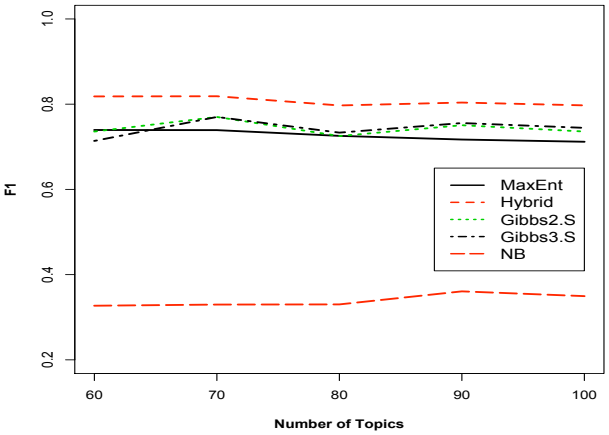

Figure 7: Performance of different methods VS. Topic number on NIPS

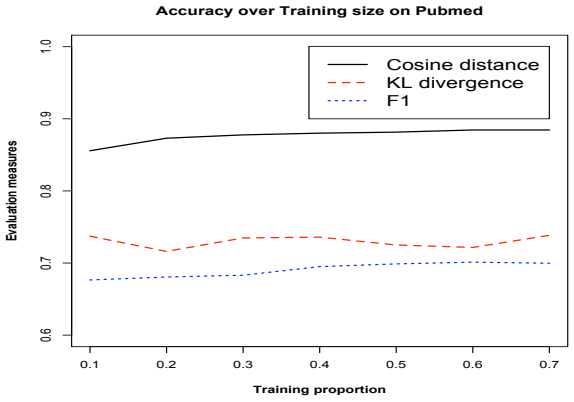

Figure 8: Performance VS. Training proportion for the light-weight MaxEnt inference method, relative to Gibbs2 on Pubmed.

as Gibbs2 does not resample topics for the training documents. These results indicate that the MaxEnt inference method provides a reasonable ordering of topics for a new document, but that if the exact topic proportions are required, two-iteration Gibbs sampling and hybrid MaxEnt should be considered.

We analyzed errors in precision and recall for the MaxEnt inference method. We discovered that for some documents, recall is high. In this case, MaxEnt can predict the major topics correctly, but it will include more topics within the 80% threshold, each with a lower weight compared to Gibbs1, Gibbs2 and Gibbs3. For some documents, precision is high. In this case, MaxEnt assigns more weight to the most prominent topics, causing other topics to fall outside the threshold. To demonstrate the comparison between topic proportions proposed by different inference methods, we show values of $\hat{\theta}$ for a single document under different inference methods in Figure 5. The proportions assigned by the MaxEnt inference method roughly match those of the sampling-based inference methods, especially compared to the naïve Bayes inference method.

We next compare the efficiency in time of each inference method. We perform experiments on a cluster of 2.66GHz Xeon X5355 processors. Figure 6 shows on a log scale that the methods can be loosely grouped into "fast" methods (classification-based, two-iteration sampling, and hybrid MaxEnt) that require a small number of passes through the data and use simple computations, and "slow" methods that require many iterations and complicated functions and 50-200 times slower.

Figure 7 shows the F1 measure of different methods given different values of T. The results indicate that the "fast" methods are relatively insensitive to the number of topics. Thus for more topics, the slow methods will be more than 100-200 times slower.

One scenario for topic model inference is in very large collections. Even with fast sampling methods, training topic models remains computationally expensive. Parallel implementations are hindered by the need for frequent interprocess communication [6]. We would like to be able to train a model on a representative subset of a collection and then estimate topic distributions for the remaining documents using light-weight MapReduce-style processing, in which every document can be handled independently. In order to study whether training size will affect prediction performance, we carried out experiment on Pubmed. We fixed 30% of the whole data as test data, and vary the training proportion from 10% to 70%. Results are shown in Figure 8 for the most light-weight inference method, MaxEnt, relative to Gibbs2. Although the size of the training set does affect performance, the magnitude of variation is relatively small.

Another application area is in streaming document collections. In this setting new documents may arrive more quickly than it is possible to train a complete model. Alternatively, we may wish to train a topic model on a central compute cluster and then "publish" a model to distributed devices that do not have the computational power to perform full topic inference, for example in an email tagging application on a smart phone. One difficulty in this setting is that the underlying topic distribution may shift over

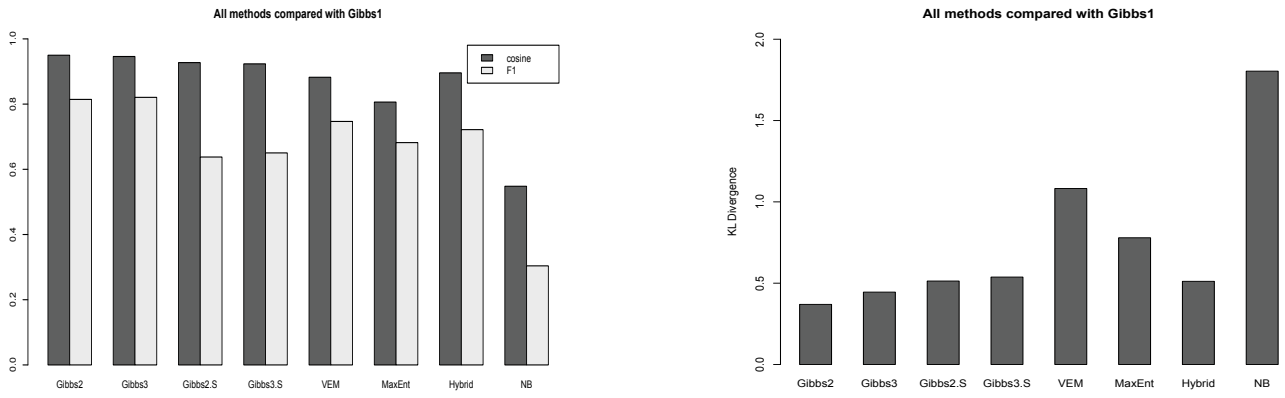

Figure 3: All methods compared with Gibbs1 on NIPS. Larger values are better in the left figure (cosine distance and F1), while smaller KL divergences are better in the right figure.

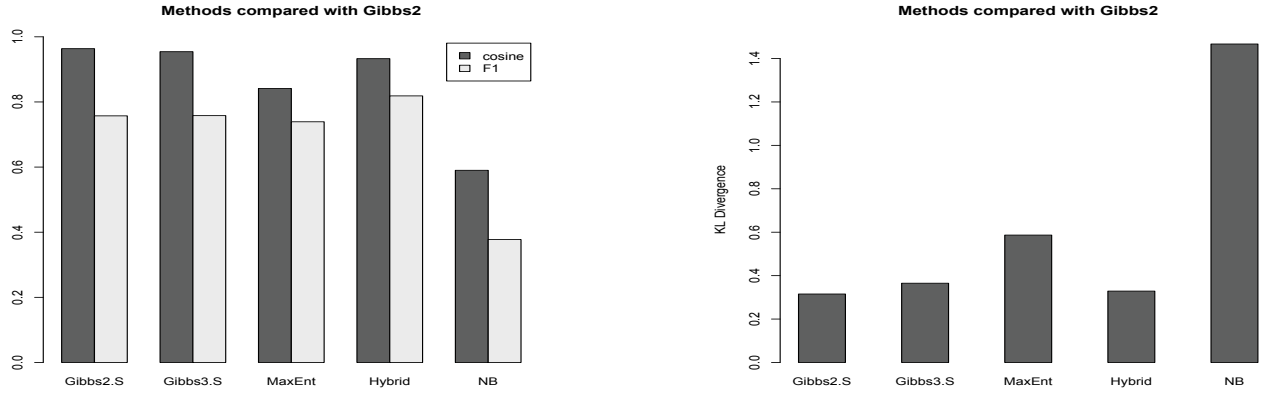

Figure 4: Methods compared with Gibbs2 on NIPS. Results for expensive iterative methods (Gibbs3, variational EM) are not shown.

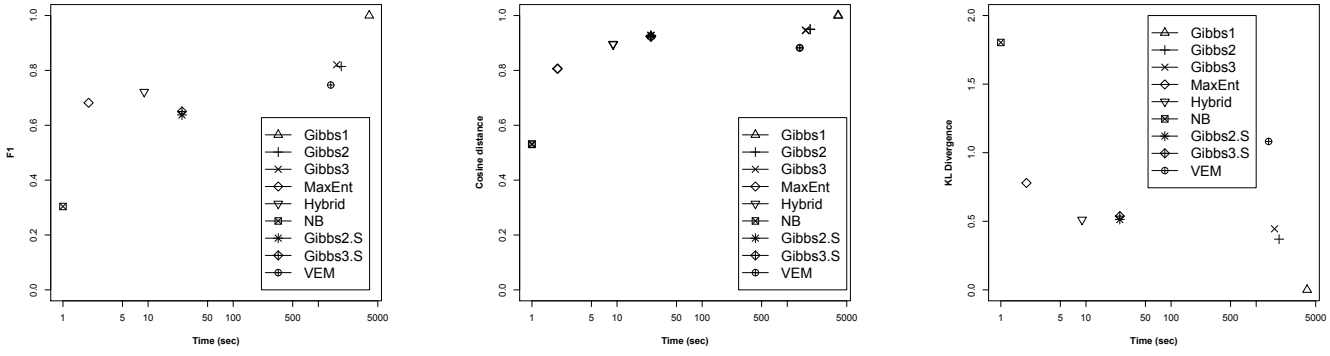

Figure 6: Performance of different methods VS. Time on NIPS

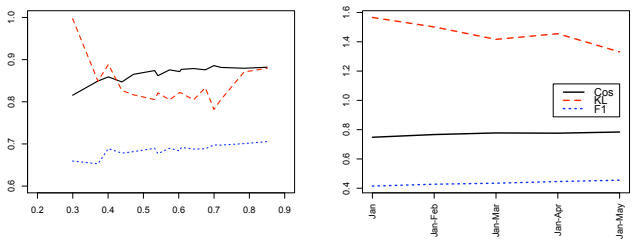

Figure 9: The effect of "topic drift" on MaxEnt performance. The left figure shows the effect of the proportion of previously seen vocabulary. The right figure shows performance on news data: we predict topics for June 2007 using only Jan 2007, then Jan and Feb, and so on.

time, rendering the trained model less applicable. One easily measured proxy for topic drift is the proportion of words that are included in the training data. We analyze how the percentage of previously seen vocabulary affects the performance of topic distribution prediction on Pubmed. On the left side of Figure 9 the x axis is the ratio of unique words seen in training data to the number of unique words in test data, and the y axis is the values of all evaluation measures. Again, we present only results for the most light-weight inference method, MaxEnt. The performance of the estimator increases as the proportion of previously seen vocabulary increases, leveling off at around 70% for cosine and F1 metrics (the KL divergence metric appears more noisy). This result suggests a simple heuristic for determining when a new model should be retrained, i.e. the training set should contain roughly 70% of the distinct word types in the "new" documents.

We also explore the effect of topic drift in news data in the New York Times data set, using June 2007 documents as test data. We use five training corpora: Jan 2007, Jan and Feb, etc. Retraining after each month improves performance, as shown on the right side of Figure 9.

7. CONCLUSIONS

Topic models provide a useful tool in analyzing complicated text collections, but their computation complexity has hindered their use in large-scale and real-time applications. Although there has been recent work on improving training-time estimation in topic models [6, 7], there has been little attention paid to the practically important problem of inferring topic distributions given existing models. Exceptions include the dynamic mixture model for online pattern discovery in multiple time-series data [10], the online LDA algorithm [8], and especially the work of Banerjee and Basu [1], who advocate the use of mixture of von Mises-Fisher distributions for streaming data based on a relatively simple document-level clustering criterion. In this paper, we focus on the more flexible topic modeling framework, which is more effective at capturing combinations of *aspects* of documents, rather than simple document similarity. Focusing on LDA, we compare a number of different methods for inferring topic distributions.

We find that a simple discriminatively trained classification-based method, MaxEnt, produces reasonable results with extremely small computational requirements. If more accuracy is required, small numbers of iterations of Gibbs sampling or the hybrid MaxEnt method provide improved results with minimal extra computation.

Finally, the efficiency of Gibbs sampling both for training topic models and inferring topic distributions for new documents can be significantly improved by the SparseLDA method proposed in this paper. This method is valuable not only because of its reduction in sampling time (at least two times the speedup reported in previous work [7]), but also because of its dramatically reduced memory usage, an issue not addressed previously.

8. ACKNOWLEDGMENTS

This work was supported in part by the Center for Intelligent Information Retrieval, in part by The Central Intelligence Agency, the National Security Agency and National Science Foundation under NSF grant #IIS-0326249, and in part by UPenn NSF medium IIS-0803847. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

9. REFERENCES

[1] A. Banerjee and S. Basu. Topic models over text streams: A study of batch and online unsupervised learning. In *SIAM-DM*, 2007.

[2] A. Berger, S. D. Pietra, and V. D. Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22:39–72, 1996.

[3] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, January 2003.

[4] T. Griffiths and M. Steyvers. Finding scientific topics. *PNAS*, 101(Suppl. 1):5228–5235, 2004.

[5] A. K. McCallum. MALLET: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[6] D. Newman, A. Asuncion, P. Smyth, and M. Welling. Distributed inference for latent dirichlet allocation. In *NIPS*, 2007.

[7] I. Porteous, D. Newman, A. Ihler, A. Asuncion, P. Smyth, and M. Welling. Fast collapsed Gibbs sampling for latent Dirichlet allocation. In *SIGKDD*, 2008.

[8] X. Song, C.-Y. Li, B. L. Tseng, and M.-T. Sun. Modeling and predicting personal information dissemination behavior. In *KDD*, 2005.

[9] Y. W. Teh, D. Newman, and M. Welling. A collapsed variational bayesian inference algorithm for latent dirichlet allocation. In *NIPS*, 2006.

[10] X. Wei, J. Sun, and X. Wang. Dynamic mixture models for multiple time series. In *IJCAI*, 2007.