Online Inference of Topics with Latent Dirichlet Allocation

Kevin R. Canini
Computer Science Division
University of California
Berkeley, CA 94720
kevin@cs.berkeley.edu

Lei Shi
Helen Wills Neuroscience Institute
University of California
Berkeley, CA 94720
lshi@berkeley.edu

Thomas L. Griffiths
Department of Psychology
University of California
Berkeley, CA 94720
tom_griffiths@berkeley.edu

Abstract

Inference algorithms for topic models are typically designed to be run over an entire collection of documents after they have been observed. However, in many applications of these models, the collection grows over time, making it infeasible to run batch algorithms repeatedly. This problem can be addressed by using online algorithms, which update estimates of the topics as each document is observed. We introduce two related Rao-Blackwellized online inference algorithms for the latent Dirichlet allocation (LDA) model – incremental Gibbs samplers and particle filters – and compare their runtime and performance to that of existing algorithms.

1 INTRODUCTION

Probabilistic topic models are often used to analyze collections of documents, each of which is represented as a mixture of topics, where each topic is a probability distribution over words. Applying these models to a document collection involves estimating the topic distributions and the weight each topic receives in each document. A number of algorithms exist for solving this problem (e.g., Hofmann, 1999; Blei et al., 2003; Minka and Lafferty, 2002; Griffiths and Steyvers, 2004), most of which are intended to be run in "batch" mode, being applied to all the documents once they are collected. However, many applications of topic models are in contexts where the collection of documents is growing. For example, when inferring the topics of news articles or communications logs, documents arrive in a continuous stream, and decisions must be made on a regular basis, without waiting for future documents to arrive. In these settings, repeatedly running a batch algorithm can be infeasible or wasteful.

In this paper, we explore the possibility of using online inference algorithms for topic models, whereby the representation of the topics in a collection of documents is incrementally updated as each document is added. In addition to providing a solution to the problem of growing document collections, online algorithms also open up different routes for parallelization of inference from batch algorithms, providing ways to draw on the enhanced computing power of multiprocessor systems, and different tradeoffs in runtime and performance from other algorithms.

We discuss algorithms for a particular topic model: latent Dirichlet allocation (LDA) (Blei et al., 2003). The state space from which these algorithms draw samples is defined at time i to be all possible topic assignments to each of the words in the documents observed up to time i. The result is a Rao-Blackwellized sampling scheme (Doucet et al., 2000), analytically integrating out the distributions over words associated with topics and the per-document weights of those topics.

The plan of the paper is as follows. Section 2 introduces the LDA model in more detail. Section 3 discusses one batch and three online algorithms for sampling from LDA. Section 4 discusses the efficient implementation of one of the online algorithms – particle filters. Section 5 describes a comparative evaluation of the algorithms, and Section 6 concludes the paper.

2 INFERRING TOPICS

Latent Dirichlet allocation (Blei et al., 2003) is widely used for identifying the topics in a set of documents, building on previous work by Hofmann (1999). In this model, each document is represented as a mixture of a fixed number of topics, with topic z receiving weight

Appearing in Proceedings of the 12^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2009, Clearwater Beach, Florida, USA. Volume 5 of JMLR: W&CP 5.

$\theta_z^{(d)}$ in document d, and each topic is a probability distribution over a finite vocabulary of words, with word w having probability $\phi_w^{(z)}$ in topic z. The generative model assumes that documents are produced by independently sampling a topic z for each word from $\theta^{(d)}$ and then independently sampling the word from $\phi^{(z)}$. The independence assumptions mean that the document is treated as a bag of words, so word ordering is irrelevant to the model. Symmetric Dirichlet priors are placed on $\theta^{(d)}$ and $\phi^{(z)}$, with $\theta^{(d)} \sim \text{Dirichlet}(\alpha)$ and $\phi^{(z)} \sim \text{Dirichlet}(\beta)$, where α and β are hyperparameters that affect the relative sparsity of these distributions. The complete probability model is thus

$$\begin{array}{lll}
w_i|z_i, \phi^{(z_i)} & \sim & \text{Discrete}(\phi^{(z_i)}), \quad i = 1, \ldots, N, \\
\phi^{(z)} & \sim & \text{Dirichlet}(\beta), \quad z = 1, \ldots, T, \\
z_i|\theta^{(d_i)} & \sim & \text{Discrete}(\theta^{(d_i)}), \quad i = 1, \ldots, N, \\
\theta^{(d)} & \sim & \text{Dirichlet}(\alpha), \quad d = 1, \ldots, D,
\end{array}$$

where N is the total number of words in the collection, T is the number of topics, D is the number of documents, and d_i and z_i are, respectively, the document and topic of the ith word, w_i. The goal of inference in this model is to identify the values of ϕ and θ, given a document collection represented by the sequence of N words $\mathbf{w}_N = (w_1, \ldots, w_N)$. Estimation is complicated by the latent variables $\mathbf{z}_N = (z_1, \ldots, z_N)$, the topic assignments of the words. Various algorithms have been proposed for solving this problem, including a variational Expectation-Maximization algorithm (Blei et al., 2003) and Expectation-Propagation (Minka and Lafferty, 2002). In the collapsed Gibbs sampling algorithm of Griffiths and Steyvers (2004), ϕ and θ are analytically integrated out of the model to collect samples from $P(\mathbf{z}_N|\mathbf{w}_N)$. The use of conjugate Dirichlet priors on ϕ and θ makes this analytic integration straightforward, and also makes it easy to recover the posterior distribution on ϕ and θ given $\mathbf{z}_N$ and $\mathbf{w}_N$, meaning that a set of samples from $P(\mathbf{z}_N|\mathbf{w}_N)$ is sufficient to estimate ϕ and θ.

Existing inference algorithms provide users with several options in trading off bias and runtime. However, most of these algorithms are designed to be run over an entire document collection, requiring multiple sweeps to produce good estimates of ϕ and θ. While some applications of these models involve the analysis of static databases, more typically, users work with document collections that grow over time. In the remainder of the paper, we outline three related algorithms that can be used for inference in such a setting.

3 ALGORITHMS

In this section, we describe a batch sampling algorithm for LDA. We then discuss ways in which this algorithm can be extended to yield three online algorithms.

Algorithm 1 batch Gibbs sampler for LDA

```
initialize $\mathbf{z}_N$ randomly from $\{1, \ldots, T\}^N$
loop
    choose $j$ from $\{1, \ldots, N\}$
    sample $z_j$ from $P(z_j|\mathbf{z}_{N\backslash j}, \mathbf{w}_N)$
```

3.1 BATCH GIBBS SAMPLER

Griffiths and Steyvers (2004) presented a collapsed Gibbs sampler for LDA, where the state space is the set of all possible topic assignments to the words in every document. The Gibbs sampler is "collapsed" because the variables θ and ϕ are analytically integrated out, and only the latent topic variables $\mathbf{z}_N$ are sampled. The topic assignment of word j is sampled according to its conditional distribution

$$P(z_j|\mathbf{z}_{N\backslash j}, \mathbf{w}_N) \propto \frac{n_{z_j,N\backslash j}^{(w_j)} + \beta}{n_{z_j,N\backslash j}^{(\cdot)} + W\beta} \frac{n_{z_j,N\backslash j}^{(d_j)} + \alpha}{n_{\cdot,N\backslash j}^{(d_j)} + T\alpha}, \quad (1)$$

where $\mathbf{z}_{N\backslash j}$ indicates $(z_1, \ldots, z_{j-1}, z_{j+1}, \ldots, z_N)$, W is the size of the vocabulary, $n_{z_j,N\backslash j}^{(w_j)}$ is the number of times word w_j is assigned to topic z_j, $n_{z_j,N\backslash j}^{(\cdot)}$ is the total number of words assigned to topic z_j, $n_{z_j,N\backslash j}^{(d_j)}$ is the number of times a word in document d_j is assigned to topic z_j, and $n_{\cdot,N\backslash j}^{(d_j)}$ is the total number of words in document d_j, and all the counts are taken over words 1 through N, excluding the word at position j itself (hence the $N\backslash j$ subscripts).

The Gibbs sampling procedure, outlined in Algorithm 1, converges to the desired posterior distribution $P(\mathbf{z}_N|\mathbf{w}_N)$. This batch Gibbs sampler can be extended in several ways, leading to efficient online sampling algorithms for LDA.

3.2 O-LDA

A simple modification of the batch Gibbs sampler yields an online algorithm presented by Song et al. (2005) and called "o-LDA" by Banerjee and Basu (2007). This procedure, outlined in Algorithm 2, first applies the batch Gibbs sampler to a prefix of the full dataset, then samples the topic of each new word i by conditioning on the words observed so far[1]:

$$P(z_i|\mathbf{z}_{i-1}, \mathbf{w}_i) \propto \frac{n_{z_i,i\backslash i}^{(w_i)} + \beta}{n_{z_i,i\backslash i}^{(\cdot)} + W\beta} \frac{n_{z_i,i\backslash i}^{(d_i)} + \alpha}{n_{\cdot,i\backslash i}^{(d_i)} + T\alpha}. \quad (2)$$

[1] The o-LDA algorithm as presented by Banerjee and Basu (2007) samples the next topic by conditioning only on the topics of the words up to the end of the previous document, rather than all previous words. The algorithm presented here is slightly slower, but more accurate.

Algorithm 2 o-LDA (initialized with first σ words)

```
sample $\mathbf{z}_\sigma$ using batch Gibbs sampler
for $i = \sigma + 1, \ldots, N$ do
    sample $z_i$ from $P(z_i|\mathbf{z}_{i-1}, \mathbf{w}_i)$
```

Algorithm 3 incremental Gibbs sampler for LDA

```
for $i = 1, \ldots, N$ do
    sample $z_i$ from $P(z_i|\mathbf{z}_{i-1}, \mathbf{w}_i)$
    for $j$ in $\mathcal{R}(i)$ do
        sample $z_j$ from $P(z_j|\mathbf{z}_{i\setminus j}, \mathbf{w}_i)$
```

After its batch initialization phase, o-LDA applies Equation (2) incrementally for each new word w_i, never resampling old topic variables. For this reason, its performance depends critically on the accuracy of the topics inferred during the batch phase. If the documents used to initialize o-LDA are not representative of the full dataset, it could be led to make poor inferences. Also, because each topic variable is sampled by conditioning only on previous words and topics, samples drawn with o-LDA are not distributed according to the true posterior distribution $P(\mathbf{z}_N|\mathbf{w}_N)$. To remedy these issues, we consider online algorithms that revise their decisions about previous topic assignments.

3.3 INCREMENTAL GIBBS SAMPLER

Extending o-LDA to occasionally resample topic variables, we introduce the incremental Gibbs sampler, an algorithm that rejuvenates old topic assignments in light of new data. The incremental Gibbs sampler, outlined in Algorithm 3, does not have a batch initialization phase like o-LDA, but it does use Equation (2) to sample topic variables of new words. After each step i, the incremental Gibbs sampler resamples the topics of some of the previous words. The topic assignment z_j of each index j in the "rejuvenation sequence" $\mathcal{R}(i)$ is drawn from its conditional distribution

$$P(z_j|\mathbf{z}_{i\setminus j}, \mathbf{w}_i) \propto \frac{n^{(w_j)}_{z_j,i\setminus j} + \beta}{n^{(\cdot)}_{z_j,i\setminus j} + W\beta} \frac{n^{(d_j)}_{z_j,i\setminus j} + \alpha}{n^{(d_j)}_{\cdot,i\setminus j} + T\alpha}. \qquad (3)$$

If these rejuvenation steps are performed often enough (depending on the mixing time of the induced Markov chain), the incremental Gibbs sampler closely approximates the posterior distribution $P(\mathbf{z}_i|\mathbf{w}_i)$ at every step i. Indeed, convergence is guaranteed as the number of times each z_j is resampled goes to infinity, since the algorithm becomes a batch Gibbs sampler for $P(\mathbf{z}_i|\mathbf{w}_i)$ in the limit. More generally, the incremental Gibbs sampler is an instance of the decayed MCMC framework introduced by Marthi et al. (2002). The choice of the number of rejuvenation steps to perform determines the runtime of the incremental Gibbs sampler.

Algorithm 4 particle filter for LDA

```
initialize weights $\omega_0^{(p)} = P^{-1}$ for $p = 1, \ldots, P$
for $i = 1, \ldots, N$ do
    for $p = 1, \ldots, P$ do
        set $\omega_i^{(p)} = \omega_{i-1}^{(p)} P(w_i|\mathbf{z}_{i-1}^{(p)}, \mathbf{w}_{i-1})$
        sample $z_i^{(p)}$ from $P(z_i^{(p)}|\mathbf{z}_{i-1}^{(p)}, \mathbf{w}_i)$
    normalize weights $\boldsymbol{\omega}_i$ to sum to 1
    if $\|\boldsymbol{\omega}_i\|^{-2} \leq$ ESS threshold then
        resample particles
        for $j$ in $\mathcal{R}(i)$ do
            for $p = 1, \ldots, P$ do
                sample $z_j^{(p)}$ from $P(z_j^{(p)}|\mathbf{z}_{i\setminus j}^{(p)}, \mathbf{w}_i)$
        set $\omega_i^{(p)} = P^{-1}$ for $p = 1, \ldots, P$
```

If $|\mathcal{R}(i)|$ is bounded as a function of i, then the overall runtime is linear. However, if $|\mathcal{R}(i)|$ grows logarithmically or linearly with i, then the overall runtime is log-linear or quadratic, respectively. $\mathcal{R}(i)$ can also be chosen to be nonempty only at certain intervals, leading to an incremental Gibbs sampler that only rejuvenates itself periodically (for example, whenever there is time to spare between observing documents).

An alternative approach to frequently resampling previous topic assignments is to concurrently maintain multiple samples of $\mathbf{z}_i$, rejuvenating them less frequently. This option is desirable because it allows the algorithm to simultaneously explore several regions of the state space. It is also useful in a multi-processor environment, since it is simpler to parallelize multiple samples – dedicating each sample to a single machine – than it is to parallelize operations on one sample. An ensemble of independent samples from the incremental Gibbs sampler could be used to approximate the posterior distribution $P(\mathbf{z}_N|\mathbf{w}_N)$; however, if the samples are not rejuvenated often enough, they will not have the desired distribution. With this motivation, we turn to particle filters, which perform importance weighting on a set of sequentially-generated samples.

3.4 PARTICLE FILTER

Particle filters are a sequential Monte Carlo method commonly used for approximating a probability distribution over a latent variable as observations are acquired (Doucet et al., 2001). We can extend the incremental Gibbs sampler to obtain a Rao-Blackwellized particle filter (Doucet et al., 2000), again analytically integrating out ϕ and θ to sample from $P(\mathbf{z}_i|\mathbf{w}_i)$. This use of particle filters is slightly nonstandard, since the state space grows with each observation.

The particle filter for LDA, outlined in Algorithm 4, updates samples from $P(\mathbf{z}_{i-1}|\mathbf{w}_{i-1})$ to generate samples from the target distribution $P(\mathbf{z}_i|\mathbf{w}_i)$ after each word w_i is observed. It does this by first generating a value of $z_i^{(p)}$ for each particle p from a proposal distribution $Q(z_i^{(p)}|\mathbf{z}_{i-1}^{(p)}, \mathbf{w}_i)$. The prior distribution, $P(z_i^{(p)}|\mathbf{z}_{i-1}^{(p)}, \mathbf{w}_{i-1})$, is typically used for the proposal because it is often infeasible to sample from the posterior, $P(z_i^{(p)}|\mathbf{z}_{i-1}^{(p)}, \mathbf{w}_i)$. However, since z_i is drawn from a constant, finite set of values, we can use the posterior, which is given by Equation (2) and minimizes the variance of the resulting particle weights (Doucet et al., 2000). Next, the unnormalized importance weights of the particles are calculated using the standard iterative equation

$$\begin{aligned}\frac{\omega_i^{(p)}}{\omega_{i-1}^{(p)}} &\propto \frac{P(w_i|\mathbf{z}_i^{(p)}, \mathbf{w}_{i-1})P(z_i^{(p)}|\mathbf{z}_{i-1}^{(p)})}{Q(z_i^{(p)}|\mathbf{z}_{i-1}^{(p)}, \mathbf{w}_i)} \qquad (4)\\ &= P(w_i|\mathbf{z}_{i-1}^{(p)}, \mathbf{w}_{i-1}). \qquad (5)\end{aligned}$$

The weights are then normalized to sum to 1. Equation (5) is designed so that after the weight normalization step, the particle filter approximates the posterior distribution over topic assignments as follows:

$$P(\mathbf{z}_i|\mathbf{w}_i) \approx \sum_{p=1}^{P} \omega_i^{(p)} 1_{\mathbf{z}_i}(\mathbf{z}_i^{(p)}), \qquad (6)$$

where $1_{\mathbf{z}_i}(\cdot)$ is the indicator function for $\mathbf{z}_i$. As $P \to \infty$, the right side converges to the left side, since $\mathbf{z}_i$ can assume only a finite number of values.

Over time, the weights assigned to particles diverge significantly, as a few particles come to provide a significantly better account of the observed data than the others. Resampling addresses this issue by producing a new set of particles that are more highly concentrated on states with high weight whenever the variance of the weights becomes large. A standard measure of weight variance is an approximation to the effective sample size, ESS $\approx \|\boldsymbol{\omega}\|^{-2}$, and a threshold can be expressed as some proportion of the number of particles, P.

The simplest form of resampling is to draw from the multinomial distribution defined by the normalized weights. However, more sophisticated resampling methods also exist, such as stratified sampling (Kitagawa, 1996), quasi-deterministic methods (Fearnhead, 2004), and residual resampling (Liu and Chen, 1998), which produce more diverse sets of particles. Residual resampling was used in our evaluations. Whenever the particles are resampled, their weights are all reset to P^{-1}, since each is now a draw from the same distribution and the previous weights are reflected in their relative resampling frequencies.

As in the resample-move algorithm of Gilks and Berzuini (2001), Markov chain Monte Carlo (MCMC) is used after particle resampling to restore diversity to the particle set in the same way that the incremental Gibbs sampler rejuvenates its samples, by choosing a rejuvenation sequence $\mathcal{R}(i)$ of topic variables to resample. The length of $\mathcal{R}(i)$ can be chosen to trade off runtime against performance, and the variables to be resampled can be randomly selected either uniformly or using a decayed distribution that favors more recent history, as in Marthi et al. (2002). While a uniform schedule visits earlier sites more overall, using a distribution that approaches zero quickly enough for sites in the past ensures that in expectation, each site is sampled the same number of times.

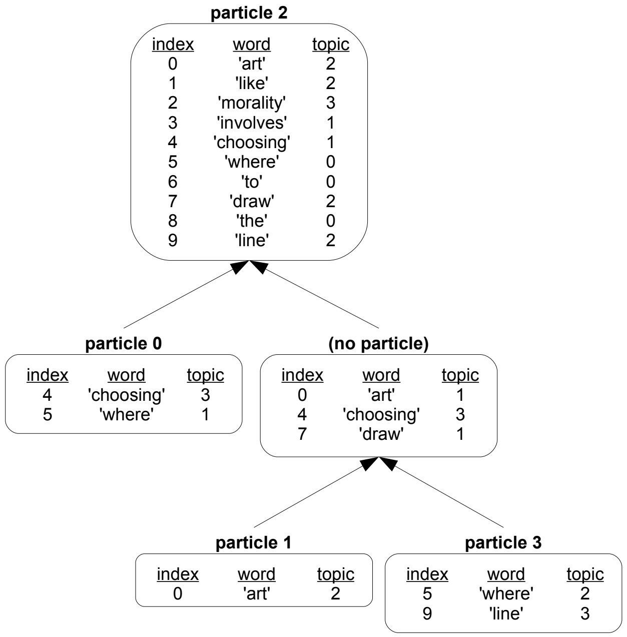
4 EFFICIENT IMPLEMENTATION

In order to be feasible as an online algorithm, the particle filter must be implemented with an efficient data representation. In particular, the amount of time it takes to incrementally process a document must not grow with the amount of data previously seen. In initial implementations of the algorithm, individual particles were represented as linear arrays of topic assignment values, consistent with the interpretation of the state space $\mathbf{z}_i = (z_1, \ldots, z_i)$ as a sequence of variables stored in an array. It was found that nearly all of the computing time was spent resampling the particles (line 8 of Algorithm 4). This is due to the fact that when a particle is resampled more than once, the naïve implementation makes copies of the $\mathbf{z}_i$ array for each child particle. Since these structures grow linearly with the observed data and resampling is performed at a roughly constant rate, the total time spent resampling particles grows quadratically.

This problem can be alleviated by using a shared representation of the particles, exploiting the high degree of redundancy among particles with common lineages. When a particle is resampled multiple times, the resulting copies all share the same parent particle and implicitly inherit its $\mathbf{z}_i$ vector as their history of topic assignments. Each particle maintains a hash table that is used to store the differences between its topic assignments and its parent's; consequently, the computational complexity of the resampling step is reduced from quadratic to linear. As illustrated in Figure 1, the particles are thus stored as a directed tree[2], with parent-child relationships indicating the hierarchy of inheritance for topic variables.

To look up the value $z_i^{(p)}$ of topic assignment i in particle p, the particle's hash table is consulted first. If the value is missing, the particle's parent's hash table is checked, recursing up the tree towards the root and

[2] More precisely, a forest of directed trees, since it is possible that not all particles share a common ancestor.



particle 2

index	word	topic
0	'art'	2
1	'like'	2
2	'morality'	3
3	'involves'	1
4	'choosing'	1
5	'where'	0
6	'to'	0
7	'draw'	2
8	'the'	0
9	'line'	2

particle 0

index	word	topic
4	'choosing'	3
5	'where'	1

(no particle)

index	word	topic
0	'art'	1
4	'choosing'	3
7	'draw'	1

particle 1

index	word	topic
0	'art'	2

particle 3

index	word	topic
5	'where'	2
9	'line'	3

Figure 1: An example of the "directed tree of hashtables" implementation of the particle filter. Particle 2 is the root, so all other particles descended from it. Particle 0 directly depends on particle 2, altering the topics of the words "choosing" and "where". The other child of particle 2 is not itself an active sample, but an inactive remnant of an old particle that was not resampled. It is retained because multiple active particles depend on its hashed topic values. If either particle 1 or particle 3 is not resampled in the future, the remaining one will be merged with its parent, maintaining the bound on the tree depth.

eventually terminating when the value is found in an ancestor's hash table. To change the value $z_i^{(p)}$, the new value is inserted into particle p's hash table, and the old value is inserted into each of particle p's children's hash tables (if they don't already have an entry) to ensure consistency.

In order to ensure that variable lookup is a constant-time operation, it is necessary that the depth of the tree does not grow with the amount of data. This can be ensured by selectively pruning the tree just after the resampling step. After resampling is performed, a node is called *active* if it has been sampled one or more times, and *inactive* otherwise. If an entire subtree of nodes is inactive, it is deleted. If an inactive node has only one active descendant depending on its history, that descendant's hash table is merged with its own. When this operation is performed after each resampling step, it can be shown that the depth of the tree is never greater than P. Furthermore, merging the hash tables takes linear amortized time. Empirical evidence confirms that the time overhead of maintaining a directed tree of hash tables is negligible compared to the increase in speed and decrease in memory usage it affords. Specifically, by reducing the resampling step to have linear runtime, this implementation detail is the key to making the particle filter feasible to run.

5 EVALUATION

In online topic modeling settings, such as news article clustering, we care about two aspects of performance: the quality of the solutions recovered and runtime. As documents arrive and are incrementally processed, we would like online algorithms to maintain high-quality inferences and to produce topic labels quickly for new documents. Since there is a tradeoff between runtime and inference quality, the algorithms were evaluated by comparing the quality of their inferences while constraining the amount of time spent per document. We compared the performance of the three online algorithms presented in Section 3: o-LDA, the incremental Gibbs sampler, and the particle filter. Our evaluation is a variation of the comparison of o-LDA to other online algorithms by Banerjee and Basu (2007), using the same datasets and performance metric.

5.1 DATASETS

The datasets used to test the algorithms are each a collection of categorized documents. They consist of four subsets derived from the 20 Newsgroups corpus[3]: (1) `diff-3` (2995 documents, 7670 word types, 3 categories), (2) `rel-3` (2996 documents, 10091 word types, 3 categories), (3) `sim-3` (2980 documents, 5950 word types, 3 categories), and (4) `subset-20` (1997 documents, 13341 word types, 20 categories), representing different levels of size and difficulty, as well as news articles harvested from the Slashdot website: (5) `slash-7` (6714 documents, 5769 word types, 7 categories) and (6) `slash-6` (5182 documents, 4498 word types, 6 categories).

5.2 METHODOLOGY

Our testing methodology is designed to approximate a real-world online inference task. For each dataset, the algorithms were given the first 10% of the documents to use for initialization. A single sample drawn using the batch Gibbs sampler on this initial set was used to initialize all of the online algorithms. This constituted the explicit batch initialization phase of o-LDA, and the other two online algorithms used the same starting configuration.

[3] Available online at `http://people.csail.mit.edu/jrennie/20Newsgroups/`

After the batch initialization set is chosen, the o-LDA algorithm has no remaining parameters and is the fastest of the three online algorithms, since it does not rejuvenate its topic assignments. The incremental Gibbs sampler has one parameter: the choice of rejuvenation sequence $\mathcal{R}(i)$. The particle filter has two parameters: the effective sample size (ESS) threshold, which controls how often the particles are resampled, and the choice of rejuvenation sequence. Since the runtime and performance of the incremental Gibbs sampler and the particle filter depend on these parameters, there is a compromise to be made. We set the parameters so that these algorithms ran within roughly 6 times the amount of time taken by o-LDA on each dataset. For the incremental Gibbs sampler, $\mathcal{R}(i)$ was chosen to be a set of 4 indices from 1 to i chosen uniformly at random. For the particle filter, the ESS threshold was set at 20 for the `diff-3`, `rel-3`, and `sim-3` datasets and at 10 for the `subset-20`, `slash-6`, and `slash-7` datasets, $|\mathcal{R}(i)|$ was set at 30 for the `diff-3`, `rel-3`, and `sim-3` datasets and at 10 for the `subset-20`, `slash-6`, and `slash-7` datasets, and the particular values of $\mathcal{R}(i)$ were chosen uniformly at random from 1 to i. The runtime of these algorithms can be chosen to fit any constraints, but we selected one point that we felt was reasonable. Indeed, an important strength of these two online algorithms is that they can take full advantage of any amount of computing power by appropriate choice of their parameters.

The LDA hyperparameters α and β were both set to be 0.1. The particle filter was run with 100 particles; to allow the other algorithms the same advantage of multiple samples, they were each run 100 times independently, with the sample having highest posterior probability at each step being used for evaluation.

Since the datasets are collections of documents with known category memberships, we evaluated how well the clustering implied by the inferred topics matched the true categories. That is, for each dataset, the number of topics T was set equal to the number of categories, and the documents were clustered according to their most frequent topic. Normalized mutual information (nMI) was used to measure the similarity of this implied partition to the true document categories (Banerjee and Basu, 2007). Scores are between 0 and 1, with a perfect match receiving a score of 1.

Two different evaluations were made for each algorithm on each dataset. First, we evaluated how well the algorithms clustered the documents on which they were trained. That is, at regular intervals throughout each dataset, the sample with maximum posterior probability was drawn, and the quality of the induced clustering of the documents observed so far was measured. Second, we evaluated how well the algorithms clustered a randomly-chosen held-out set consisting of 10% of the documents, as a function of the amount of the training set that had been observed so far. That is, at regular intervals throughout the training set, each algorithm was run on the held-out documents as if they were the next ones to be observed, the nMI score was calculated for the held-out documents, and the algorithm was returned to its original state and position in the training set.

5.3 RESULTS

The results of the training set evaluation are shown in Figure 2. The particle filter and incremental Gibbs sampler perform about equally well, with the particle filter performing better for some datasets. As expected, o-LDA consistently has the lowest score of the three algorithms. The dashed horizontal line in each figure represents the performance of the batch Gibbs sampler on the entire dataset, which is approximately the best possible performance an online algorithm could achieve using the LDA model.

The results of the evaluation on the held-out set are shown in Figure 3. For each held-out set, the mean performance of the particle filter is consistently better than that of the incremental Gibbs sampler, which is consistently better than that of o-LDA. With the exception of the `sim-3` and `subset-20` datasets, the algorithms' performances are separated by at least two standard deviations. Interestingly, the performance of all the algorithms on all the held-out document sets does not change significantly as more training data is observed. This seems to indicate that a majority of the information about the topics comes from the first 10% of the documents. In `rel-3`, performance on the held-out set seems to decrease as more of the training set is observed. This could be because the held-out documents are more closely related to those at the beginning of the training set than those at the end.

As mentioned earlier, the algorithms' performance strongly depends on their parameters; allowing more time for rejuvenation of old topic assignments would improve the performance of the particle filter and the incremental Gibbs sampler. Table 1 summarizes the total runtimes of each algorithm on each dataset. Although there is some variation, the incremental Gibbs sampler and particle filter each take about 6 times longer than o-LDA.

The top ten words from 5 of the 20 topics found by the particle filtering algorithm on the `subset-20` dataset are listed in Table 2. Although the normalized mutual information is not as high as that of the batch Gibbs sampler, the recovered topics seem intelligible.

We also noticed that the particle filter used signifi-

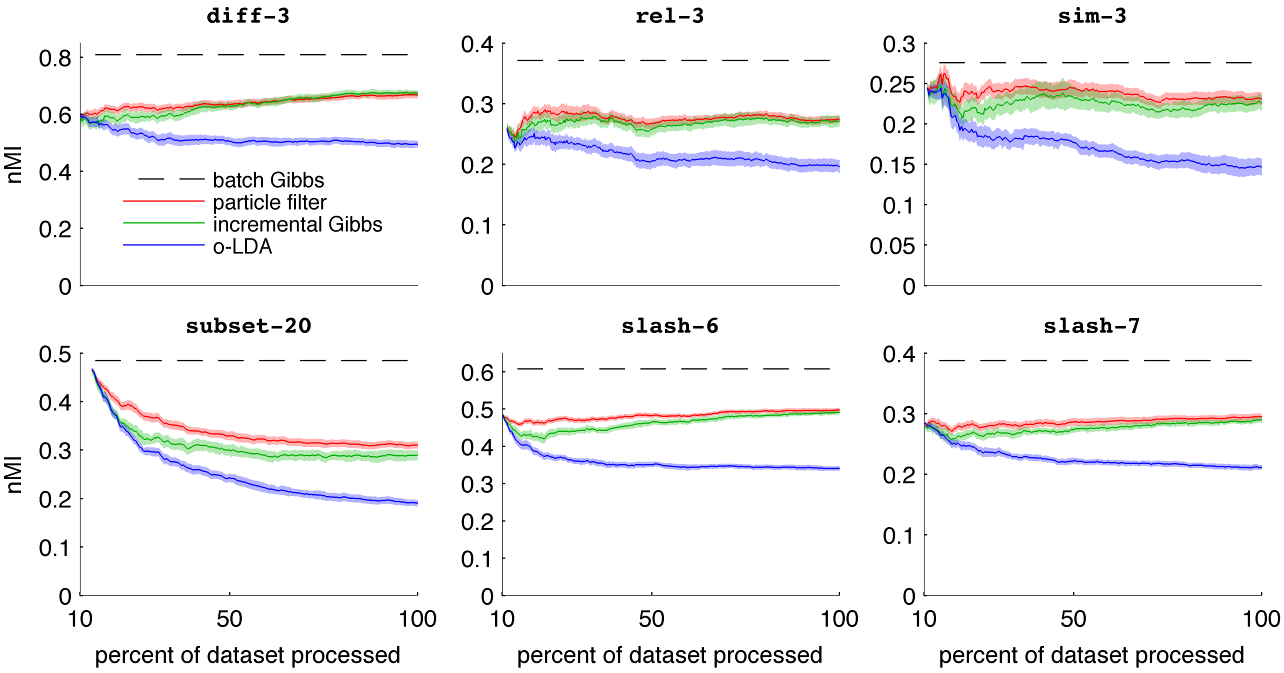


Figure 2: nMI traces for each algorithm on each dataset. The algorithms were initialized with the same configuration on the first 10% of the documents. Each dashed horizontal line represents the nMI score for the batch Gibbs sampler on an entire dataset. Solid lines show mean performance over 30 runs, and shading indicates plus and minus one sample standard deviation.

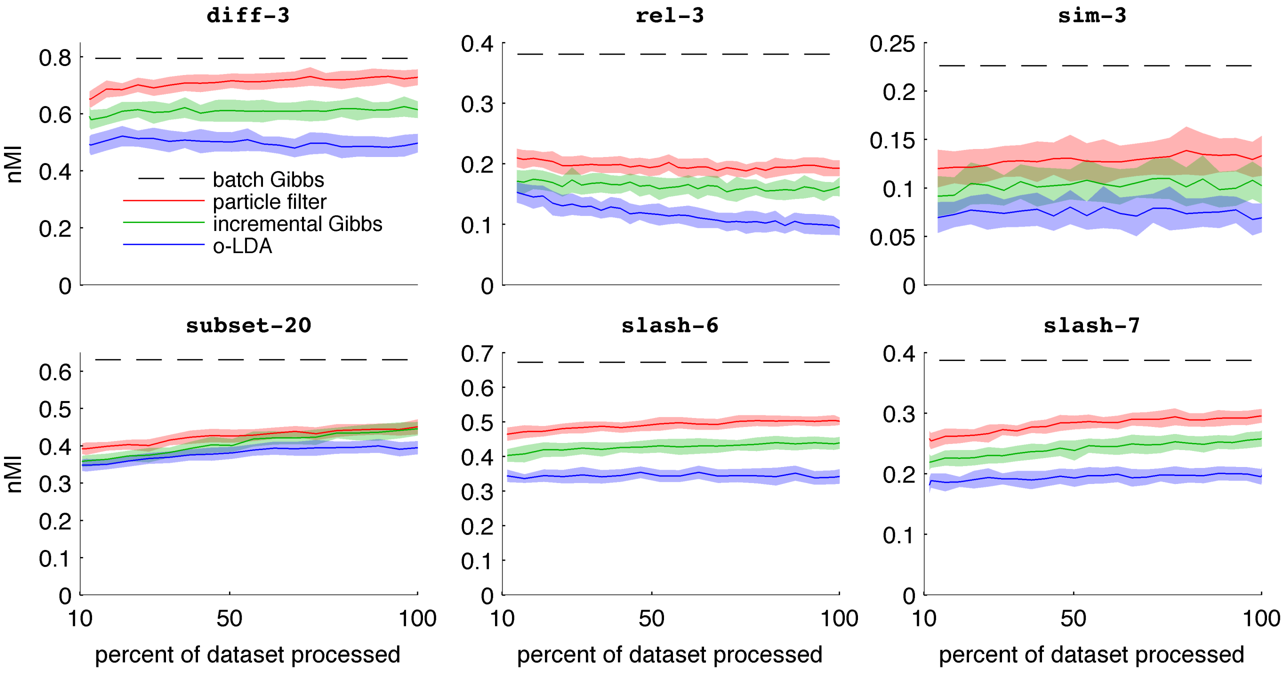


Figure 3: nMI traces for each algorithm on held-out test sets, as a function of the amount of the training set observed. The algorithms were initialized with the same configuration on the first 10% of the documents. Each dashed horizontal line represents the nMI score on the held-out set for the batch Gibbs sampler given the entire training set. Solid lines show mean performance over 30 runs, and shading indicates plus and minus one sample standard deviation.

Table 1: Runtimes of algorithms in seconds. Numbers in parentheses give multiples of o-LDA runtime.

	o-LDA	inc. Gibbs	particle filter
`diff-3`	34	185 (5.4)	183 (5.4)
`rel-3`	57	338 (5.9)	251 (4.4)
`sim-3`	32	176 (5.5)	148 (4.6)
`subset-20`	150	1221 (8.1)	1029 (6.9)
`slash-6`	44	255 (5.8)	256 (5.8)
`slash-7`	69	420 (6.1)	521(7.6)

Table 2: The top 10 words from 5 topics found by the particle filter for the `subset-20` dataset.

gun	list	space	god	wire
police	mail	cost	bible	ground
guns	users	nasa	moral	wiring
semi	email	mass	choose	cable
fire	internet	rockets	christian	power
cops	access	station	jesus	neutral
revolver	unix	orbit	church	nec
carry	address	launch	christianity	circuit
auto	security	dod	absolute	box
koresh	system	drink	life	current

cantly less memory than the other online algorithms. This is due to the shared representation of the particles, as discussed in Section 4, which allows more accurate inferences to be computed with less memory than algorithms that maintain independent samples.

6 DISCUSSION

We have discussed a number of algorithms for the problem of inferring topics using LDA. We have shown how to extend a batch algorithm into a series of online algorithms, each more flexible than the last. Our results demonstrate that these algorithms perform effectively in recovering the topics used in multiple datasets. Latent Dirichlet allocation has been extended in a number of directions, including incorporation of hierarchical representations (Blei et al., 2004), minimal syntax (Griffiths et al., 2004), the interests of authors (Rosen-Zvi et al., 2004), and correlations between topics (Blei and Lafferty, 2005). We anticipate that the algorithms we have outlined here will naturally generalize to many of these models. In particular, applications of the hierarchical Dirichlet process to text (Teh et al., 2006) can be viewed as an analogue to LDA in which the number of topics is allowed to vary. Our particle filtering framework requires little modification to handle this case, providing an online alternative to existing inference algorithms for this model.

Acknowledgements

This work was supported by the DARPA CALO project and NSF grant BCS-0631518. The authors thank Jason Wolfe for helpful discussions and Sugato Basu for providing the datasets and nMI code used for evaluation.

References

Banerjee, A. and Basu, S. 2007. Topic models over text streams: a study of batch and online unsupervised learning. In *Proc. 7th SIAM Int'l. Conf. on Data Mining.*

Blei, D. M., Griffiths, T. L., Jordan, M. I., and Tenenbaum, J. B. 2004. Hierarchical topic models and the nested Chinese restaurant process. In *Advances in Neural Information Processing Systems 16.*

Blei, D. M. and Lafferty, J. D. 2005. Correlated topic models. In *Advances in Neural Information Processing Systems 18.*

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

Doucet, A., de Freitas, N., and Gordon, N., eds. 2001. *Sequential Monte Carlo Methods in Practice.* Springer.

Doucet, A., de Freitas, N., Murphy, K. P., and Russell, S. J. 2000. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *Proc. 16th Conf. in Uncertainty in Artificial Intelligence.*

Fearnhead, P. 2004. Particle filters for mixture models with an unknown number of components. *Statistics and Computing*, 14(1):11–21.

Gilks, W. R. and Berzuini, C. 2001. Following a moving target–Monte Carlo inference for dynamic Bayesian models. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 63(1):127–146.

Griffiths, T. L. and Steyvers, M. 2004. Finding scientific topics. *Proc. National Academy of Sciences of the USA*, 101(Suppl. 1):5228–5235.

Griffiths, T. L., Steyvers, M., Blei, D. M., and Tenenbaum, J. B. 2004. Integrating topics and syntax. In *Advances in Neural Information Processing Systems 17.*

Hofmann, T. 1999. Probabilistic latent semantic indexing. In *Proc. 22nd Annual Int'l. ACM SIGIR Conf. on Research and Development in Information Retrieval.*

Kitagawa, G. 1996. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25.

Liu, J. S. and Chen, R. 1998. Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1032–1044.

Marthi, B., Pasula, H., Russell, S. J., and Peres, Y. 2002. Decayed MCMC filtering. In *Proc. 18th Conf. in Uncertainty in Artificial Intelligence.*

Minka, T. P. and Lafferty, J. D. 2002. Expectation-propagation for the generative aspect model. In *Proc. 18th Conf. in Uncertainty in Artificial Intelligence.*

Rosen-Zvi, M., Griffiths, T. L., Steyvers, M., and Smyth, P. 2004. The author-topic model for authors and documents. In *Proc. 20th Conf. in Uncertainty in Artificial Intelligence.*

Song, X., Lin, C.-Y., Tseng, B. L., and Sun, M.-T. 2005. Modeling and predicting personal information dissemination behavior. In *Proc. 11th ACM SIGKDD Int'l. Conf. on Knowledge Discovery and Data Mining.*

Teh, Y. W., Jordan, M. I., Beal, M. J., and Blei, D. M. 2006. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581.