A recursive estimate for the predictive likelihood in a topic model

James G. Scott
McCombs School of Business
and Division of Statistics and Scientific Computing
University of Texas at Austin

Jason Baldridge
Department of Linguistics
University of Texas at Austin

Abstract

We consider the problem of evaluating the predictive log likelihood of a previously unseen document under a topic model. This task arises when cross-validating for a model hyperparameter, when testing a model on a hold-out set, and when comparing the performance of different fitting strategies. Yet it is known to be very challenging, as it is equivalent to estimating a marginal likelihood in Bayesian model selection. We propose a fast algorithm for approximating this likelihood, one whose computational cost is linear both in document length and in the number of topics. The method is a first-order approximation to the algorithm of Carvalho et al. (2010a), and can also be interpreted as a one-particle, Rao-Blackwellized version of the "left-to-right" method of Wallach et al. (2009). On our test examples, the proposed method gives similar answers to these other methods, but at lower computational cost.

1 INTRODUCTION

Topic models are a popular technique for performing dimensionality reduction on large, unstructured text corpora. The central idea of topic models is that documents can be represented as a weighted mixture of latent themes, which are in turn represented as weighted lists of words. An appealing property of topics is that they can typically can be viewed semantically, as a simple characterization of an idea or concept. This makes them useful for exploratory data analysis of large text collections, identifying connections between documents, and more.

Appearing in Proceedings of the 16^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2013, Scottsdale, AZ, USA. Volume 31 of JMLR: W&CP 31.

Topic models originated with Latent Semantic Analysis (Landauer and Dumais, 1997). Blei et al. (2003) later provided a probabilistic formulation with their Latent Dirichlet Allocation (LDA) model. Since then, topic models have become a fundamental tool in the analysis of large, unstructured bodies of text, especially because the LDA framework can be extended to incorporate additional dependencies, such as time, geography and authorship (see Blei (2012) for an overview). Yet it is very difficult to estimate predictive likelihoods (equivalently, perplexity) on held-out documents under even the basic LDA model, and many methods that have been proposed for doing so it are insufficient and/or incorrect. Wallach et al. (2009) provide an excellent summary of past work on this issue. Their particle-Gibbs ("left-to-right") method does correctly approximate the held-out likelihood, but it involves an expensive $O(N^2)$ resampling step, where N is the number of words in a document.

We propose to estimate the held-out likelihood using a sequential Monte Carlo algorithm called particle learning, or PL (Carvalho et al., 2010a). Particle learning is a recent proposal for handling parameter uncertainty in state-space models—a domain very different from the present context—and does not seem to be in wide use by the natural language processing community (one exception is Sales et al., 2012). Nonetheless, it turns out to be ideally suited for estimating the predictive likelihood of a topic model.

Ultimately, we recommend a fast, deterministic analogue of particle learning, involving a moment-based approximation to the filtered distribution for the PL sufficient statistics. We describe the full PL approach and compare it with a Gibbs sampling approach, and then describe the filtering-based approximation of PL. Our empirical results suggest that the approximation offers a favorable trade of accuracy for speed in estimating held-out likelihoods. Additionally, our results show that permuting word order and considering different draws of the topic distributions from the posterior produce greater variation for a given evaluation

method compared to the variation between methods.

To fix notation: let $y_i \in \{1, \ldots, D\}$ be the ith word in a document, D be the dictionary size, and K the number of topics. (Note that the y_i are not word counts, but rather dictionary indices corresponding to tokens. For example, if $y_1 = 10$, the first word in the document is the 10th word in the dictionary.) Let B denote the $D \times K$ topic matrix whose kth column (denoted B_k) represents the multinomial distribution over words associated with the kth topic: $B_k = (B_{1k}, \ldots, B_{Dk})^T$. The document-specific topic loadings $f = (f_1, \ldots, f_K)$ are then modeled as a draw from the prior distribution $p(f \mid \alpha)$. For the sake of illustration, we assume throughout that this corresponds to the LDA model, although neither the issues we raise nor the solution we propose depend in any important way on this choice of model. Under this assumption, the marginal distribution for the ith word is the mixture

$$p(y_i = d \mid f) \propto \sum_{k=1}^{K} f_k B_{d,k} \,,$$

and the prior for f is $f \sim \text{Dirichlet}(\alpha_1, \ldots, \alpha_K\}$.

We do not discuss the actual model-fitting step here. For this we refer the reader to the discussion in Taddy (2012), who compares various strategies based on Markov-chain Monte Carlo, variational EM, and joint MAP estimation of B and f for all documents.

2 SEQUENTIAL MONTE CARLO

2.1 The need for approximation

Suppose we fit a topic model to a corpus, giving us point estimates for the topics B and the hyperparameter α. (In some cases α is not estimated, but merely fixed in the prior.) The fitting may be done by whatever method, so long as a point estimate for these quantities may be extracted.

Now we observe a new document y, which we recall is represented as a vector of dictionary indices $(y_1, \ldots, y_N)$, $y_i \in \{1, \ldots, D\}$. We wish to compute the marginal likelihood for this unseen document, conditioning on the estimated corpus-level parameters, $p(y \mid B, \alpha)$. This is a painful computation involving a high-dimensional integral over the unknown vector of topic weights for the new document:

$$p(y \mid B, \alpha) \quad = \quad \int_{\Delta_K} p(y \mid B, f)\; p(f \mid \alpha)\; df \,,$$

Here Δ_K is the K-dimensional simplex, and $p(f \mid \alpha)$ is the Dirichlet prior for the topic weights.

Such integrals are called marginal likelihoods (a.k.a. evidence or predictive likelihoods) in Bayesian inference, and arise routinely in model selection (e.g. Berger and Pericchi, 2001). They are notoriously difficult to compute outside simple conjugate families. The difficulty is even more severe in the present case, as $p(y \mid B, f)$ is itself a mixture distribution that is usually represented in terms of another high-dimensional integral over latent allocations of words to topics. For models of this kind, there is often no simple way to proceed (Basu and Chib, 2003).

The calculation of such integrals by computationally intensive means is very much an active research area in Bayesian inference, as well as in statistical physics (e.g. Skilling, 2006). These methods simply do not scale to the typical topic-modeling problem, where many different models or model-fitting strategies might be entertained, and where a held-out set might consist of thousands of documents or more. Since the marginal likelihood must be calculated for each document, fast approximations will inevitably be necessary.

There is a very different reason why an approximation to the held-out likelihood is the best that can be expected. In most applications of topic modeling, there is a major pre-processing step required, whereby stop words are filtered out. These are typically high-frequency function words, such as determiners, prepositions, and some adverbs. Without stop word filtering, a "vanilla" LDA topic model is forced to account for such words, and—due to their high frequency across all documents—they end up polluting all estimated topics. These words are ignored when computing the likelihood; this means that the likelihood computations are not only throwing away information, but they are in fact throwing away a large portion of the document—typically about half. The bias introduced by using $p(y \mid B, \alpha)$ as an estimate of $p(y, y_{stop} \mid B, \alpha)$ is likely large compared to the inaccuracies of any particular technique for calculating $p(y \mid B, \alpha)$.

Further considerations in computing likelihood with topic models come with both word order and different samples (point estimates) of the topics. In our experiments, we compute likelihoods for different draws from the posterior of B and different word order permutations, for several evaluation methods. The results show that both of these factors introduce greater within-method variability than across-method variability—so much that the different methods are essentially indistinguishable from one another as good estimators of the likelihood. So, we can only hope for an approximation at best when we compute likelihood using a point estimate and a single word order.

2.2 Particle learning

Before getting to our deterministic approximation, we first describe how the topic-model marginal likelihood may be approximated using particle learning (PL). Our approach is closest to that of Carvalho et al. (2010b) and Merl (2009), who study the use of PL in nonparametric Bayesian density estimation. In the time series literature, "filtering" means tracking an underlying state variable that changes in time (like the position of a satellite observed with error). "Learning" means estimating the fixed parameters of a hidden-variable model as data arrives. Particle learning is one such method. It re-casts learning about parameters as the problem of filtering for the sufficient statistics corresponding to those parameters.

Let y_i denote the ith word in the document, and let $y^{(i)}$ denote the collection of words $\{y_1, \ldots, y_i\}$. We factorize the joint likelihood as a product of conditionals:

$$p(y \mid B, \alpha) = p(y_1 \mid B, \alpha) \cdot \prod_{i=2}^{N} p(y_i \mid B, \alpha, y^{(i-1)}) \,. \quad (1)$$

Importantly, this factorization holds for any ordering. Each term in the product may be written as

$$p(y_i \mid B, \alpha, y^{(i-1)}) = \int_{\Delta_K} p(y_i \mid B, f)\, p(f \mid B, \alpha, y^{(i-1)}) \,. \quad (2)$$

We recognize $p(f \mid B, \alpha, y^{(i-1)})$ as the conditional posterior for the topic weights f, given words 1 through $i-1$ in the new document. If this conditional posterior were a Dirichlet distribution, then the integral in (2), the one-step-ahead predictive likelihood, would be available in closed form. To see this, let γ_i be the latent topic indicator for word i, and observe that

$$p(y_i = d \mid B, f) = \sum_{k=1}^{K} p(\gamma_i = k \mid f) \cdot p(y_i = d \mid \gamma_i = k, B) \,.$$

The integral of $p(\gamma_i = k \mid f)$ may be explicitly calculated with respect to a Dirichlet prior, and the second term inside the sum is known. Letting $w_{ik} = \alpha_k + z_{ik}$,

$$p(y_i = j \mid B, \alpha, z_i, y^{(i-1)}) \;=\; \sum_{k=1}^{K} \left(\frac{w_{ik}}{\sum_l w_{il}} \right) B_{j,k} \,.$$

This would allow us to calculate (1) term by term.

Of course, because $p(f \mid B, \alpha, y^{(i-1)})$ is not a Dirichlet distribution, this simple argument fails. But it is a mixture of Dirichlets. Introduce the latent variable $z_i = (z_{i1}, \ldots, z_{iK})$, a vector whose kth entry indicates how many words have been allocated to topic k from words 1 through $i-1$. This latent z_i is a sufficient statistic for the parameter f_i, in the sense that

$$\begin{aligned} p(y_i \mid B, \alpha, z_{i-1}, y^{(i-1)}) &= p(y_i \mid B, \alpha, z_{i-1}) \\ p(f \mid \alpha, z_{i-1}, y^{(i-1)}) &= p(f \mid \alpha, z_{i-1}) \,. \end{aligned} \quad (3)$$

Moreover,

$$(f \mid \alpha, z_i, y^{(i-1)}) \;\sim\; \mathrm{Dir}(w_{i1}, \ldots, w_{iK}) \,, \quad (4)$$

exploiting standard results on the conjugate Dirichlet-multinomial family. Therefore, $p(f \mid B, \alpha, y^{(i)})$ may be represented in terms of the sufficient statistic z_i:

$$p(f \mid \alpha, y^{(i-1)}) = \int_{\mathcal{Z}} p(f \mid B, \alpha, z)\; p(z \mid B, y^{(i-1)})\; dz \,.$$

The step-i conditional posterior for the sufficient statistic z therefore carries all the information necessary to compute (1) term by term. In particle learning, we represent this distribution using a discrete, or particle, approximation: $p(z \mid y^{(i-1)}) \approx \frac{1}{M} \sum_{t=1}^{M} \delta_{z^{(t)}}$ for sufficiently large M, with δ_z representing a Dirac measure at the point z. In this way, particle learning turns a smoothing problem for the latent allocations $\gamma_1, \ldots, \gamma_i$ into a filtering problem for the *aggregated counts* of allocations $z_1, \ldots, z_K$. This reduction to sufficient statistics is an important step: $y^{(i)}$ grows in dimensionality as we accumulate data, whereas z_i remains of fixed dimension K.

Moreover, we may exploit a simple recursion for updating our knowledge about the essential state vector z, given the next word y_i:

$$\begin{aligned} p(z_i \mid y^{(i)}) &= \int_{\mathcal{Z}} p(z_i \mid z_{i-1}, y_i)\; p(z_{i-1} \mid y^{(i)})\; dz_{i-1} \\ p(z_{i-1} \mid y^{(i)}) &\propto p(z_{i-1} \mid y^{(i-1)})\; p(y_i \mid z_{i-1}) \,. \end{aligned} \quad (5)$$

This factorization shows that our discrete approximation may be updated from $p(z \mid y^{(i-1)})$ to $p(z \mid y^{(i)}) = p(z \mid y^{(i-1)}, y_i)$ in two steps. Let Z_{i-1} be the set of particles representing our uncertainty about the essential state vector, up through word $i-1$.

(1) Resample the particles in Z_{i-1} with replacement M times, with weights π_t proportional to the one-step-ahead predictive likelihood for particle $z_{i-1}^{(t)}$. This corresponds to the second line of (5). Supposing that $y_i = j$, the jth word in the dictionary, these weights may be calculated from (4) as

$$\pi_t = p(y_i = j \mid z_{i-1}^{(t)}) = \sum_{k=1}^{K} \left\{ \frac{\alpha_k + z_{i-1,k}^{(t)}}{\sum_l (\alpha_l + z_{i-1,l}^{(t)})} \right\} B_{j,k} \,.$$

This will generate a new set Z_i of M particles, including many ties from the old particle set Z_{i-1}.

Algorithm 1 PL for LDA predictive likelihood

RESAMPLE(Z, Π, M): an algorithm to resample the elements of vector Z with replacement M times, with sampling weights Π.
B: a $D \times K$ matrix of topics.
α: the Dirichlet hyperparameter.
M: number of particles.
Z_0 : particles $z_0^{(t)} = (0, \ldots, 0)_K$ for $t = 1, \ldots, M$.
$l \leftarrow 0$
for $i = 1$ to number of words **do**
  Observe $y_i = j$.
  for $t = 1$ to M **do**
    $\pi^{(t)} \leftarrow \sum_{k=1}^{K} \left\{ \frac{\alpha_k + z_{i-1,k}^{(t)}}{\sum_l (\alpha_l + z_{i-1,l}^{(t)})} \right\} B_{j,k}$
  end for
  $\Pi \leftarrow \{\pi^{(1)}, \ldots, \pi^{(M)}\}$
  $l \leftarrow l + \log \left\{ M^{-1} \sum_{t=1}^{M} \pi^{(t)} \right\}$
  $Z_i = \text{RESAMPLE}(Z_{i-1}, \Pi, M)$
  for $t = 1$ to M **do**
    Sample $d_i^{(t)} = k \in \{1, \ldots, K\}$ where

$$p(d_i^{(t)} = k) = \left\{ \frac{\alpha_k + z_{i,k}^{(t)}}{\sum_l (\alpha_l + z_{i,l}^{(t)})} \right\} B_{j,k}$$

    $z_{i,k}^{(t)} \leftarrow z_{i,k}^{(t)} + 1$.
  end for
end for
OUTPUT: l, the estimated log likelihood.

(2) Propagate each particle in Z_i according to the first line of (5). Do this by drawing a particle-specific topic assignment $d_i^{(t)} \in \{1, \ldots, K\}$ for the new word, where

$$p(d_i^{(t)} = k \mid z_i^{(t)}, y_i = j) \propto \left\{ \frac{\alpha_k + z_{i,k}^{(t)}}{\sum_l (\alpha_l + z_{i,l}^{(t)})} \right\} B_{j,k}\,,$$

again supposing that $y_i = j$. Letting k denote the sampled topic, increment $z_{ik}^{(t)}$ by one, and leave all other entries of $z_i^{(t)}$ unchanged.

After applying both the resampling and propagation steps, one is left with a particle cloud Z_i that approximates $p(z \mid B, y^{(i)})$.

The full particle-learning algorithm (Algorithm 1) involves initializing a set of M (say, 10^4) particles with empty state vectors $z^{(t)}, t = 1, \ldots, M$, and then recursively applying these resample/propagate steps as each word y_i is processed. Although the goal of PL is to track $p(z \mid y^{(i)})$, the filtered posterior distribution over the sufficient statistics, the algorithm also provides an estimate for the predictive likelihood (1) as a by-product. This is because each contribution to the likelihood (2) can be approximated as

$$\begin{aligned} p(y_i \mid B, \alpha, y^{(i-1)}) &\approx \hat{L}_i = \frac{1}{M} \sum_{t=1}^{M} L_i^{(t)} \\ L_i^{(t)} &= \sum_{k=1}^{K} \left(\frac{\alpha_k + z_{i-1,k}^{(t)}}{\sum_l \alpha_l + z_{i-1,l}^{(t)}} \right) B_{y_i,k}\,. \end{aligned}$$

The overall log likelihood for the new document can be estimated by carrying along the sum of the $\log \hat{L}_i$ terms as the algorithm proceeds.

2.3 Comparison with Gibbs sampling

Particle learning involves the usual trick in any sequential Monte Carlo algorithm, which is to represent our uncertainty about an unknown quantity using a particle approximation. In spirit, it is therefore similar to the "left-to-right" Gibbs-sampling algorithm of Wallach et al. (2009). This algorithm represents uncertainty in $p(f \mid B, \alpha, y^{(i-1)})$ by introducing the latent topic allocations of words 1 to $i-1$. As before, denote this set by $\gamma^{(i-1)}$. Given $\gamma^{(i-1)}$, the conditional posterior for the topic weights is Dirichlet. Wallach et al. (2009) then marginalize over uncertainty in $\gamma^{(i-1)}$ by representing $p(\gamma^{(i-1)} \mid y^{(i-1)})$ using a particle cloud (say of size M), where each particle has its own complete set of allocations $\gamma^{(i-1)}$. Every time a new word is processed, three actions are taken within each particle: (1) a single Gibbs-sampling pass is made over the previous indicators $(\gamma_1, \ldots, \gamma_{i-1})$; (2) the predictive likelihood $p(y_i \mid \gamma^{(i-1)}, B)$ is calculated, using the newly updated indicators, and added to the running total; and (3) a topic indicator for word i is sampled from $p(\gamma_i \mid y_i, B, \gamma^{(i-1)})$. Thus the number of latent allocations that must be sampled is $O(MN^2)$, where N is the document length.

It is evident that step 1 (backwards re-sampling of indicators) is necessary for the particle average in step 2 to be integrating over the correct distribution: $p(\gamma_1, \ldots, \gamma_i \mid B, y^{(i)})$. Approximating this smoothed distribution is an extremely difficult task, because the dimension of the parameter vector grows precisely at the rate at which data y_i are accumulated. In PL, by contrast, the M particles do not represent uncertainty over the full set of topic allocations. Rather, they represent uncertainty over conditional sufficient statistics z_i for that document's topic weights. This can be estimated more easily, for the simple reason that there is much less information to track. (Obviously if we knew the smoothed posterior distribution $p(\gamma_1, \ldots, \gamma_i \mid B, y^{(i)})$, then the conditional posterior of z would be known trivially. But the converse is not true.) This also reduces the computational complexity of the algorithm to $O(MN)$, rather than $O(MN^2)$.

To avoid the $O(N^2)$ resampling step, it is tempting to forego the resampling step within the left-to-right algorithm. That is, once word i is assigned to a specific topic, that allocation remains fixed (within each particle) as new words are processed. We refer to this as the no-Gibbs left-to-right algorithm. This brings the cost of the method back down to $O(MN)$, at the cost of a severely degraded approximation. We strongly recommend against this approach, as explained below.

2.4 Sources of error in each approximation

There are many potential sources of systematic error in each of these methods. (By this we mean errors that are distinct from the randoms errors introduced by any Monte Carlo method.) A problem common to both methods is that we are approximating the individual terms in the product (1) and multiplying together the results. The errors therefore also multiply, rather than add up. This will be a problem with any sequential algorithm; it is difficult to see how it can be avoided without using Gibbs sampling (e.g. Chib, 1995) or some more general-purpose, computationally intensive method. For the reasons already described, these are infeasible given the scale of our data sets and the limits of current computing technology. Thus the bias introduced by the product approximation is almost certainly something we will have to live with.

In particle learning, another possible source of error is the degradation of the particle approximation to $p(z \mid y^{(i-1)})$ as more data is accumulated. This issue is refereed to as "particle decay" in the sequential Monte Carlo literature, and refers to the possibility that most of the information content in the estimate will be carried by a small handful particles. The particle-decay issue is discussed at great length in Carvalho et al. (2010a), as well as the paper by Lopes et. al. together with the many published discussions of this paper in the Valencia 9 proceedings (Lopes et al., 2011). It is impossible to summarize these lengthy discussions here. But it seems fair to say that there are many examples of models where particle learning seems to work well compared with MCMC, and at least a few where it does not. The empirical results of Section 4 suggest that the former is more likely for topic models, although given our inability to compute a "ground truth" answer for a high-dimensional marginal likelihood, it is impossible to know this for sure.

As for the left-to-right algorithm, recall the logic of the Gibbs-sampling step: in order for the particle cloud to give a good approximation to $p(y_i \mid y^{(i-1)}, B)$, each particle must represent a draw from the smoothed posterior over the full state vector, $p(\gamma^{(i-1)} \mid B, y^{(i-1)})$. A single Gibbs pass over the previous indicators defines a transition operator that maps the current state to a new state that accounts for the new data point. But this one-step operator may not lead to mixing that is sufficiently rapid for the final vector to represent a draw from the correct joint distribution. Even if these errors introduced by poor mixing are small, they may accumulate within a particle, and cannot be solved with more particles. This is in some sense the analogue of the particle-decay problem in PL.

The main issue with the left-to-right algorithm, however, is the N^2 computational burden. The no-Gibbs left-to-right algorithm alleviates this burden, but is highly fraught. Consider one particle's state variables $\gamma^{(i-1)} = (\gamma_1, \ldots, \gamma_{i-1})$ at step i of the algorithm. If we do not resample each state variable γ_j $(j < i)$ when word i is observed and a new state variable γ_i is sampled, then this particle will not represent a draw from step-i smoothed posterior $p(\gamma^{(i)} \mid y^{(i)}, B)$. Instead, each γ_j will be a draw from an approximation to the step-j *filtered* posterior $p(\gamma_j \mid y^{(j)}, B)$. Moreover, even this approximation to each filtered posterior will degrade as more words are observed. Collectively, the particles will fail to represent the correct joint distribution over states, and the corresponding errors in approximating $p(f \mid y^{(i-1)}, B)$ may accumulate.

Another way of seeing the problem is to recognize the no-Gibbs left-to-right algorithm as, essentially, a version of particle learning where particles are never resampled. But the re-sampling step in PL is necessary to maintain an approximation to the correct posterior, meaning that the no-Gibbs left-to-right algorithm cannot be doing the right thing. We therefore strongly recommend against eliminating the resampling step.

3 APPROXIMATION BY FILTERING

The proposed particle-learning algorithm for LDA is simple enough in structure that it suggests a faster deterministic analogue. Specifically, we explore what happens when we avoid approximating $p(z \mid y^{(i-1)})$ by a particle cloud, and instead approximate it using only its first moment $E(z \mid y^{(i-1)})$. This is similar to Kalman filtering, where we try to track the conditional expected value of a hidden state variable.

Specifically, we make the following approximation:

$$\begin{aligned} p(y_i \mid B, \alpha, y^{(i-1)}) &= \int_{\Delta_K} p(y_i \mid B, f)\; p(f \mid \alpha, y^{(i-1)}) \\ &\approx \int_{\Delta_K} p(y_i \mid B, f)\; p(f \mid \alpha, \hat{z}_{i-1})\,, \end{aligned}$$

where $\hat{z}_{i-1} = E(z \mid y^{(i-1)})$ is the conditional expected value for the state vector z, given $y^{(i-1)}$. This expected value can be updated recursively using Algorithm 2.

Algorithm 2 A filtering approximation to PL

B: a $D \times K$ matrix of topics.
α: the Dirichlet hyperparameter.
M: number of particles.
$z_0 \leftarrow (0, \ldots, 0)_K$
$l \leftarrow 0$
for $i = 1$ to number of words **do**
  Observe $y_i = j$.
  for $k = 1$ to number of topics **do**
    $w_k \leftarrow \left\{ \frac{\alpha_k + z_{i-1,k}}{\sum_l (\alpha_l + z_{i-1,l})} \right\} B_{j,k}$
  end for
  $S \leftarrow \sum_{k=1}^{K} w_k$
  $l \leftarrow l + \log S$
  for $k = 1$ to number of topics **do**
    $w_k \leftarrow w_k / S$
    $z_{i,k} \leftarrow z_{i-1,k} + w_k$
  end for
end for
OUTPUT: l, the estimated log likelihood.

This first-moment-only approximation undoubtedly leads to a loss of accuracy in estimating the predictive likelihood, especially for words that are processed early on. In documents whose length is substantially less than the number of topics, the bias introduced may be substantial. We do not claim that this procedure is more accurate than a Gibbs-sampling approach. We merely argue that it is faster and leads to a sensible approximation in large-data scenarios to which MCMC methods are ill-suited. We also argue that it is a better choice than the no-Gibbs left-to-right algorithm. Moreover, our experiments suggest that any bias introduced is not severe compared with the Monte Carlo variance of existing procedures, or with the variance introduced by the additional sources mentioned in Section 2.

An equivalent way of describing our approximation is as a Rao-Blackwellized, single particle version of the no-Gibbs left-to-right algorithm. That is, we use a single particle; we accumulate *expected* topic assignments for each word, rather than sampling hard assignments; and we do not revisit past words. The Rao-Blackwellization (i.e. carrying expected counts rather than actual indicators) is key in maintaining a good approximation to $p(f \mid y^{(i-1)}, B)$, especially for words visited near the beginning of the algorithm.

4 EXPERIMENTS

Our working assumption is that the full N^2 resampling step in the left-to-right algorithm is computationally infeasible for a large corpus. If one has the resources to pay that cost, and the need for a very accurate answer, then it is probably worth running both the full PL algorithm and the full left-to-right algorithm with resampling, to compare the answers. A full study of the relative merits of these (computationally expensive) algorithms is clearly an area where further study is needed, but is beyond the scope of this paper.

Here, we focus on the filter-based approximation and the no-Gibbs left-to-right algorithm. We provide two sets of experiments on simulated data and one set on text corpora. For the latter, topics are estimated using MALLET[1]. Both MALLET's implementation of the left-to-right algorithm and our own are used.[2]

4.1 A smaller simulated example

We first conducted a small-scale simulation to assess our approximate filter-based method against three other algorithms: full PL, the no-Gibbs left-to-right algorithm, and importance sampling, a form of Monte Carlo integration that is widely used to compute marginal likelihoods in smaller-dimensional problems. Our goal was to understand the variability in the likelihood estimates caused by the choice of method, versus the variability introduced by three of the other factors mentioned in Section 2: Monte Carlo error, the effect of the order in which words are processed, and the effect of using a single draw from the posterior to represent the topic parameters $B_1, \ldots, B_K$.

We a simulated a topic model with a $D = 1000$ word dictionary, $K = 20$ topics, and documents of length $N = 250$ (roughly the length of a short news article). Topics B_k were drawn from a symmetric Dirichlet(0.05), while document-level topic weights were drawn from a Dirichlet(α), where each element α_k was drawn from a standard log-normal. We conducted four different versions of the same experiment, focusing on a single held-out document y.

1. Fix the estimate of B to its true value. Fix an order in which to process words. Estimate the held-out likelihood using each algorithm 25 times.
2. Fix the estimate of B to its true value. Estimate the held-out likelihood using each algorithm 25 times, each time with a different order of the words in y.
3. Fix an order in which to process words. Let $\hat{B}$ be a draw from the posterior over B, assuming we have observed 500 documents all simulated from the true model. Estimate the held-out likelihood using each algorithm 25 times, each time using a different draw $\hat{B}$.
4. Estimate the held-out likelihood using each algo-

[1] `http://mallet.cs.umass.edu/`

[2] For code, data and instructions for reproducing our results, see `https://github.com/utcompling/topicmodel-eval`

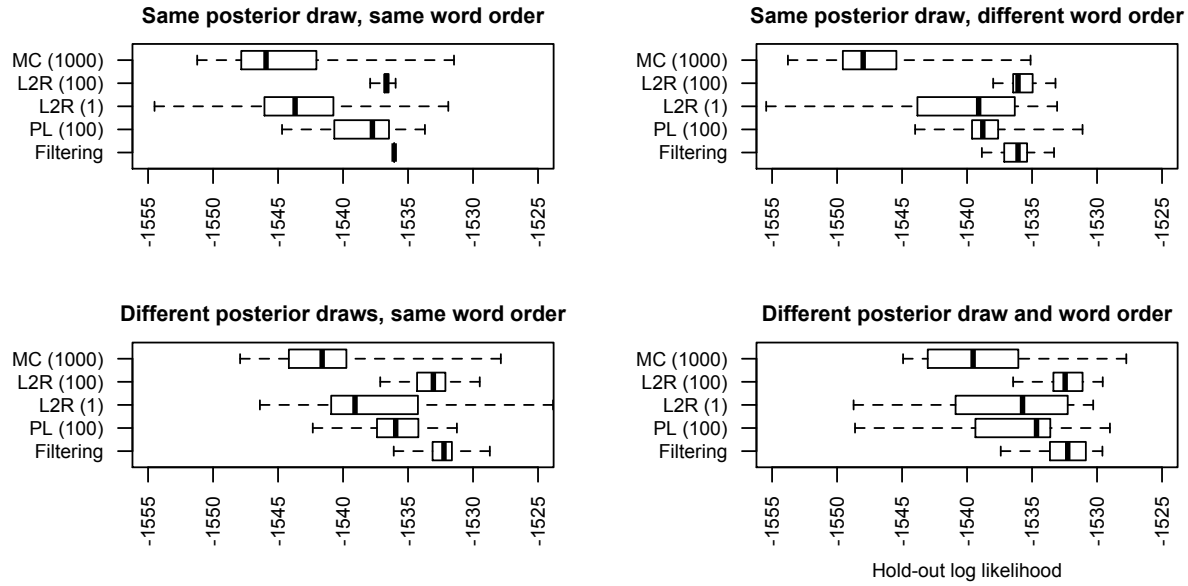

Figure 1: Results from experiments described in §4.1. L2R: left-to-right algorithm without (Gibbs) resampling. PL: particle learning. Filtering: the filter-based approximation from §3. MC: Monte Carlo integration using draws from the prior. The number in parentheses is the number of particles or Monte Carlo draws used.

rithm 25 times, each time using a different word order and a different posterior draw $\hat{B}$.

These results are shown in Figure 1. When both the posterior draw and the word order are fixed (upper left), there appear to be moderate differences, on the order of 1% on a log scale, between the average answer of each method. The approximate filtering method and the no-Gibbs left-to-right algorithm (with 100 particles) are very similar in the degree to which they approximate full PL, but the filtering method does so at 1% of the computational cost.

Moreover, when both word-order variability and posterior-draw variability are introduced, the differences between the methods look relatively smaller, compared to the variability within each method. This supports our earlier claim that the error introduced by using a fast approximate method to estimate the predictive likelihood may be swamped by other sources of variation in the entire text-processing pipeline.

4.2 Larger simulated examples

We next simulated data from a topic model of a more realistic size: a dictionary of $D = 10000$ words, $N = 1000$ held-out documents, a constant document length of $N = 500$, and $K \in \{50, 200\}$ topics. Hyperparameter settings were the same as above. This time we simulated 10 different data sets from the same model, to get a sense of how the different approximations compare. Figure 2 shows that, although there

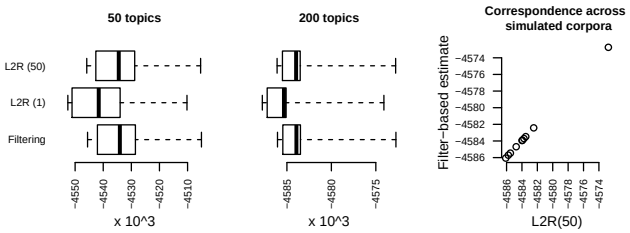

Figure 2: Comparing left-to-right (L2R, with 50 particles or one) and filtering algorithms on larger simulated data. The left and middle plots show variation in likelihood estimates for 10 different datasets for each method (using 50 and 200 topics, respectively), and the right plot shows correlation in estimates between L2R with 50 particles and filtering across the 10 simulated corpora with 200 topics.

are small and detectable differences between the filter-based approximation and the no-Gibbs left-to-right algorithm, the sampling distribution of the model estimates themselves dwarfs all other forms of variability.

4.3 Real corpora

Finally, we tested the approximate evaluation methods on the six English corpora given below.

1. **gutenberg:** 678 public domain books (published between 1798 and 2008) from Project Gutenberg.
2. **pcltravel** Ninety-four travel books from the late

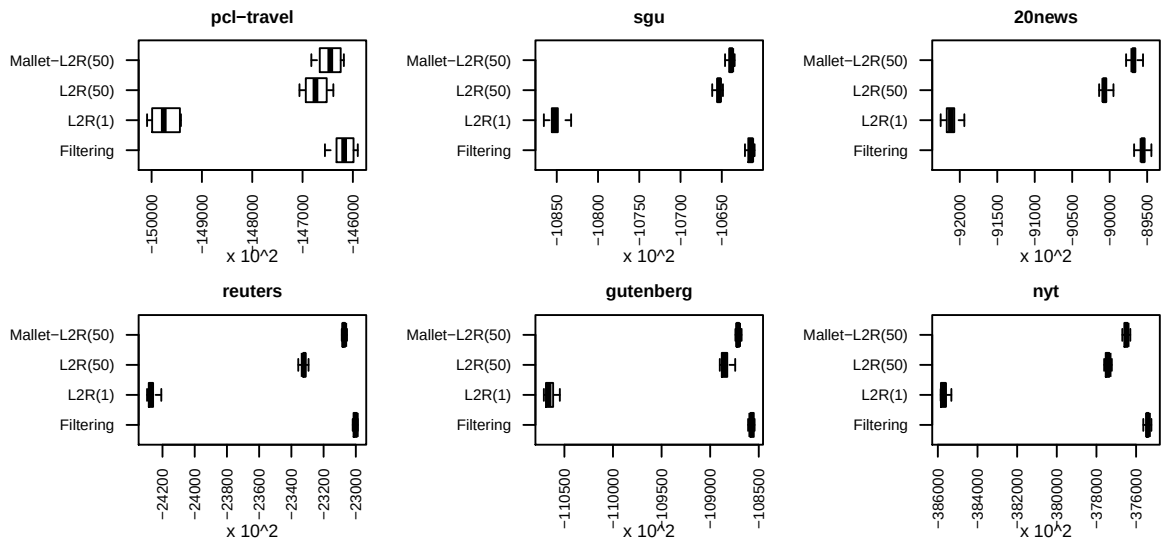

Figure 3: Results from experiments described in §4.3. L2R: left-to-right algorithm without (Gibbs) resampling. Mallet L2R: the same algorithm as implemented in MALLET.

20th century digitized by the Perry–Castañeda Library at the University of Texas at Austin.

3. **sgu**: Transcripts from 139 episodes of the *Skeptics Guide to the Universe* podcast.
4. **20news** The 20 Newsgroups data set.
5. **reuters:** The Reuters-21578 Text Categorization Collection.
6. **nyt** New York Times articles from a subset of the 2002 Gigaword corpus.

These provide a diverse mix of domains and time periods: fiction books (**gutenberg**), non-fiction books (**pcltravel**), transcribed multi-individual speech (**sgu**), web forums (**20news**), and newswire text (**reuters** and **nyt**). Full details for each corpus, including raw data for all except **nyt** and example topics computed from each, are given at the repository mentioned in footnote 2.

The **gutenberg** and **pcltravel** books were split into sub-documents of several paragraphs each as input to MALLET. The **sgu** episodes were split by show segment. Raw document boundaries were retained for the others. Table 1 provides several measurements of each corpus based on this splitting.

We ran each algorithm ten times, with a different draw from the posterior used to estimate B each time (100 topics). Figure 3 shows that there are slightly more differences between the methods here than for the simulated data. However, the differences in the average answers given by each method are not severe, and are of roughly the same order of magnitude as the variation within each method caused by having different draws from the posterior over the topic parameters. The advantage of the filtering method is that it has the same computational cost as a one-particle version of the no-Gibbs left-to-right algorithm, yet it consistently gives answers that are more in line with 50-particle version.

DATA	**V**	**N**	$\bar{\mathbf{N}}$	**StdDev**
gutenberg	78556	2953834	377	55.5
pcl-travel	188765	4780051	469	367.5
sgu	26851	421621	472	678.3
20news	114547	2743124	145	353.4
reuters	43153	1528617	70	47.9
nyt	182942	21836689	405	209.6

Table 1: Corpus statistics, aggregated over both training and held-out evaluation for each data set and ignoring stop words. V is the vocabulary size, N the number of tokens, $\bar{\mathrm{N}}$ the average document length, and StdDev the standard deviation in document length.

We attempted to make our implementation of L2R as close as possible to that in MALLET, and there are many possible explanations for the minor discrepancies seen in Figure 3. For example, the default in MALLET is to ignore unseen words in evaluating held-out likelihoods. Our implementation takes these words into account; we had to supply MALLET with the vocabulary (but not the documents) in the evaluation set to ensure they were part of the likelihood computation. This brought the likelihoods much closer, though differences remain. We merely note that the difference in estimated likelihoods between the filter-based approach and L2R(50) is roughly the same as the difference between the two implementations of L2R(50).

References

S. Basu and S. Chib. Marginal likelihood and Bayes factors for Dirichlet process mixture models. *Journal of the American Statistical Association*, 98(461): 224–35, 2003.

J. O. Berger and L. Pericchi. Objective Bayesian methods for model selection: introduction and comparison. In *Model Selection*, volume 38 of *Institute of Mathematical Statistics Lecture Notes – Monograph Series*, pages 135–207. Beachwood, 2001.

D. Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 2:993–1022, 2003.

C. M. Carvalho, M. S. Johannes, H. F. Lopes, and N. G. Polson. Particle learning and smoothing. *Statistical Science*, 25(1):88–106, 2010a.

C. M. Carvalho, H. F. Lopes, N. G. Polson, and M. A. Taddy. Particle learning for general mixtures. *Bayesian Analysis*, 5(4):709–40, 2010b.

S. Chib. Marginal likelihood from the Gibbs output. *Journal of the American Statistical Association*, 90 (432):1313–21, 1995.

T. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101:5228–35, 2004.

T. K. Landauer and S. T. Dumais. A solution to plato's problem: the latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, 104(2):211–240, 1997.

H. Lopes, C. M. Carvalho, M. Johannes, and N. G. Polson. Particle learning for sequential Bayesian computation (with discussion). In *Bayesian Statistics 9*. Oxford University Press, 2011.

D. Merl. Advances in Bayesian model based clustering using particle learning. Technical Report LLNL-TR-421078, Lawrence Livermore National Laboratory, 2009.

A. Sales, C. Challis, P. R., and D. Merl. Semisupervised classification of texts using particle learning for probabilistic automata. In *Bayesian Theory and Applications*. Oxford University Press, 2012.

J. Skilling. Nested sampling for general Bayesian computation. *Bayesian Analysis*, 1(4):833–60, 2006.

M. Taddy. On estimation and selection for topic models. In *AISTATS*, 2012.

H. Wallach, I. Murray, R. Salakhutdinov, and D. Mimno. Evaluation methods for topic models. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1105–1112. ACM, 2009.