FINAL REPORT: TOPIC MODELING ON LITERARY TEXTS

EMILY EISNER, JOE FUTOMA, AND DAVID RICE

1. Introduction

Probabilistic topic models are well known unsupervised learning algorithms, primarily used to uncover the main themes that underly a large corpus of documents. Topic models have also been applied to other types of data, including finding patterns in genetic data, images, video, and social networks [1]. However, they are usually used on text data, as the notion of a "topic" is clearly defined for text data; namely, a "topic" is some probability distribution over a fixed vocabulary of words, often with some obvious thematic relation between the most probable words.

The problem motivating our project is finding out what happens when topic models are run on literary texts. Initially, we planned to run topic models on legal texts in an effort to organize them according to underlying themes, but this data proved too difficult to obtain and preprocess. However, we obtained access to a subset of the collected texts of the Project Gutenberg corpus, which is the largest single collection of electronic free books [2]. Topic models are frequently applied to nonfiction works of text, but much less frequently applied to works of fiction and other literature. For instance, [3] applied topic models to Wikipedia, the journal *Nature*, and a large corpus of *New York Times* articles, and [4] models articles from the journal *PNAS*. Such nonfiction works are natural candidates for topic models because of the obvious information retrieval tasks that journal (and news) articles present. For instance, it is an important problem to be able to automatically organize and search massive databases of online scientific articles without the need to manually tag articles with keywords. This will allow scientists to easily find other articles relating to their discipline without the need to read many irrelevant articles.

Problems like this are not as well posed for other types of text data, however. For example, while it is straightforward to label a scientific article by its main topics, organizing fiction in such a way is not as obvious. One person's interpretation of a novel may differ dramatically from someone else's, and the themes that pervade works of fiction are less concrete and more general than the topics that exist in nonfiction articles. We want to run topic models on works of fiction, like the Gutenberg corpus, to see what types of topics exist in works of fiction. As more books become digitized, the problem of automatically sorting and organizing works of fiction in large databases becomes more relevant. Finally, although beyond the scope of this work, an interesting problem would be to model the evolution of literary themes over time, perhaps with a time-dependent topic model, and to see if this agrees with the trends in literature that literary scholars have observed in the past.

In this project, we implement and run the simplest topic model, called Latent Dirichlet Allocation (LDA), in order to analyze the latent thematic structure of the texts in our subset of the Gutenberg corpus. We first implement batch mean field variational inference for LDA, first introduced in [5], using synthetic data. However, a problem with this inference algorithm is its inability to scale to massive data. To remedy this, we implement an online learning algorithm for LDA, as described in [6]. This inference algorithm can be interpreted as a stochastic natural gradient ascent on the variational objective, and we will summarize the details below. Finally, since topic models are unsupervised learning algorithms (since there is no "true" set of topics characterizing a document), it can be difficult to quantitatively evaluate their performance. However, several methods in the

literature have been proposed to asses the performance of topic models. To this end, we also aim to implement one or two quantitative evaluation metrics for our topic models, following the work of [7].

The remainder of our report is structured as follows. In Section 2, we explain in further detail the generative process underlying LDA. We then highlight the important points from the derivations of batch mean field variational inference and online variational inference. We conclude Section 2 with a brief discussion of perplexity, our method of quantitative evaluation, described in work by [7]. In Section 3, we discuss the data we used for our implementations, describe our implementations, code for preprocessing and consulted external software. Finally in Section 4, we discuss our results and in Section 5 give a brief conclusion.

2. Methods

2.1. **Latent Dirichlet Allocation.** Probabilistic topic models are generative probabilistic models used to model discrete data, which in the scope of our work is limited to text data. Latent Dirichlet Allocation, introduced in [5], specifies a simple generative process assumed to have generated the observed data, and the problem then becomes that of inferring the appropriate parameters for this process.

First we define some common notation and highlight important assumptions underlying the model. We assume a fixed vocabulary of V unique words, and consider a corpus of D of documents. Within each document, LDA assumes *exchangeability* in the words, so that each document may be represented as a "bag or words", meaning that the order of words in the document is not considered. Additionally, the order of the documents within the corpus is also assumed to be exchangeable. Note that the assumption of exchangeability allows us to apply de Finetti's representation theorem [8]. This asserts that an infinitely exchangeable sequence of random variables are independent and identically distributed, conditioned on some random parameter, which in this case will be latent variables in the model. Thus, assuming exchangeability in the words and documents allows the joint probability distribution of the model to factor nicely into products of *i.i.d.* conditional distributions.

A key assumption of LDA is that there exists some fixed number of latent topics underlying the data, where a topic is defined as a probability distribution over the words in the vocabulary. In the simple version of LDA we consider, the number of topics is fixed a priori, although relaxing this assumption leads to more flexible Bayesian nonparametric models, as in [9]. We follow the common notation used and let K denote the fixed number of topics in the corpus, and let β_k indicates the k topic, so that $\beta_{1:K}$ or simply β refers to the collection of topics. Additionally, for each document in the corpus, $d \in \mathcal{D}$, we allow the document to exhibit multiple topics in varying proportions. Hence, all of the documents share the same set of K topics, but each exhibit these topics differently in unique proportions. For example, a document about computational neuroscience might exhibit topics related to computation and biology, while a paper about Bayesian statistics might exhibit topics about computation and inference.

We define for each document a distribution over topics θ_d to denote the mixture weights of the topics for document d. Then, we imagine that each document in the corpus is created by first choosing a distribution over the topics, θ_d. Next, for each word in the document, we choose the topic, $z_{d,n}$ that this word came from, given θ_d, and finally, given that topic, we choose a word, $w_{d,n}$. Having defined this generative process then, the goal of LDA is to compute the posterior distribution via Bayes' rule, i.e. the distribution of the hidden topics, topic proportions, and topic indicators that underly the set of documents given the observed data.

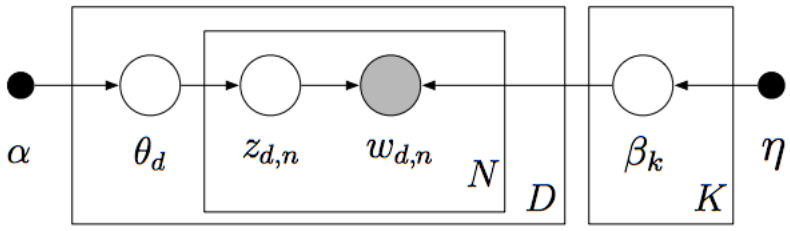

FIGURE 1. LDA Graphical Model

The generative process that defines LDA can be expressed mathematically by the following process:

(1) Draw topics $\beta_k \sim Dir_V(\eta)$ for $k \in \{1, \ldots, K\}$
(2) For each document $d \in \{1, \ldots, D\}$:
    (a) Draw topic proportions $\theta_d \sim Dir_K(\alpha)$
    (b) For each word $w \in \{1, \ldots, N_d\}$:
        (i) Draw topic assignment $z_{d,n} \sim Mult(\theta_d)$
        (ii) Draw word $w_{d,n} \sim Mult(\beta_{z_{dn}})$

We may also express this process for LDA graphically, in the form of a graphical model. Graphical models are diagrams that visually depict the conditional dependence between variables in a model. Observe the graphical model for LDA in Figure 1, from [3].

Note that this may be condensed into a single equation: the joint probability distribution of the observed data and hidden variables in the model, as follows (note that we overload notation on p):

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}|\alpha, \eta) = \prod_{k=1}^{K} p(\beta_k|\eta) \prod_{d=1}^{D} p(\theta_d|\alpha) \prod_{n=1}^{N_d} p(z_{d,n}|\theta_d) p(w_{d,n}|\beta_{z_{dn}}, z_{d,n}). \tag{1}$$

The posterior distribution, which is what we are interested in, can thus be expressed, via Bayes' rule as:

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D}|w_{1:D}) = \frac{p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D})}{p(w_{1:D})} \tag{2}$$

In most cases, the evidence term $p(w_{1:D})$in this equation is difficult to compute, since we can rewrite it in terms of the joint after integrating and summing out the hidden variables:

$$p(w_{1:D}) = \int_{\theta} \int_{\beta} \sum_{z} p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}). \tag{3}$$

Note these integrals and sums are intractable to compute, so we cannot compute the posterior directly. To address this problem of posterior inference, there are two main classes of inference algorithms. In Monte Carlo methods, we approximate the posterior by attempting to draw samples from it, in algorithms such as Markov Chain Monte Carlo (MCMC), importance sampling, Metropolis-Hastings, or many others. In our work, we instead refer to a class of algorithms known as variational inference methods.

2.2. **Mean Field Batch Variational Inference.** The idea behind so-called variational methods is to posit a simpler "variational" distribution, dependent on certain parameters, and then to optimize those parameters to be as close to the true posterior as possible. Here closeness between

distributions is measured in terms of KL Divergence, a non-symmetric metric between probability distributions.

The first variational inference algorithm we used, and implemented, is known as batch mean field variational inference. Batch refers to the fact that one iteration of the algorithm requires iteration over the entire batch of documents in the corpus, while mean field refers to an assumption we make on our variational distribution, q. Let $q(\theta, z, \beta|\gamma, \phi, \lambda)$ denote our variational distribution in question, where the θ depend on variational parameters γ, the z depend on ϕ and the β depend on λ. The mean field assumption assumes conditional independence between the hidden variables in the variational distribution. That is, the mean field assumption assumes we can factorize the variational distribution for the corpus as:

$$q(\theta, \mathbf{z}, \beta|\gamma, \phi, \lambda) = \prod_{k=1}^{K} q(\beta_k|\lambda_k) \prod_{d=1}^{D} q(\theta_d|\gamma_d) \prod_{n=1}^{N_d} q(z_{d,n}|\phi_{d,n}). \tag{4}$$

In effect, the mean field assumption has created a simpler distribution that completely factorizes by removing the coupling that exists in the true model between θ, z, and β. Note that the particular distributions we choose for each q in the above equation are chosen to be the same as the corresponding conditionals specified by LDA, i.e. Dirichlet and Multinomial.

Recall that our goal is to minimize the goal is to minimize the KL Divergence from q to p, $KL(q(\theta, \mathbf{z}, \beta)||p(\theta, \mathbf{z}, \beta|\mathbf{w}))$. It is easily verified, as in [3] that minimizing this KL Divergence is equivalent to maximizing a lower bound on the log marginal probability of the given observations, $p(\mathbf{w})$, via an application of Jensen's Inequality. We call this lower bound the Evidence Lower BOund (ELBO):

$$\log\, p(\mathbf{w}|\alpha, \eta) \geq \mathcal{L}(\mathbf{w}, \phi, \gamma, \lambda) \equiv \mathbb{E}_q[\log\, p(\mathbf{w}, \mathbf{z}, \theta, \beta|\alpha, \eta)] - \mathbb{E}_q[\log\, q(\mathbf{z}, \theta, \beta|\phi, \gamma, \lambda)]. \tag{5}$$

Note that the ELBO consists of two terms, an expectation of the log of the joint probability of the model, with respect to the variational distribution q, and an entropy term for q. Using our factorization of the joint and of the variational distribution, the ELBO may be decomposed into a summation over the documents of several smaller terms, as detailed explicitly in [6]. This observation will prove important when we explain the online version of this algorithm in the next subsection.

In order to optimize the variational parameters, we take the gradient of the ELBO with respect to each of the variational parameters, set to 0, and solve for the corresponding update equations in closed form [3, 6]. Omitting the derivation, the closed form updates for the variational parameters are given by:

$$\phi_{dwk} \propto \exp\{\mathbb{E}_q[\log\, \theta_{dk}] + \mathbb{E}_q[\log\, \beta_{kw}]\} \tag{6}$$

$$\gamma_{dk} = \alpha + \sum_{w} n_{dw}\phi_{dwk} \tag{7}$$

$$\lambda_{kw} = \eta + \sum_{d} n_{dw}\phi_{dwk} \tag{8}$$

where n_{dw} is the number of times word w appears in document d. Thus, the only information we need from the raw data is the word counts in order to implement and run the algorithm. We optimize the variational parameters via coordinate ascent, fixing all parameters but one and updating the parameter in question via the equations above. These equations are guaranteed to converge to a stationary point of the ELBO [6]. Note that the required expectations above can be

computed directly as:

$$\mathbb{E}_q[\log\ \theta_d k] = \Psi(\gamma_{dk}) - \Psi(\sum_{i=1}^{K} \gamma_{di}) \tag{9}$$

$$\mathbb{E}_q[\log\ \beta_{kw}] = \Psi(\lambda_{kw}) - \Psi(\sum_{i=1}^{V} \lambda_{ki}) \tag{10}$$

where Ψ is the digamma function (first derivative of the logarithm of the gamma function). The derivation for this is in [5] and relies on writing the Dirichlet distribution in its exponential form and then using a well known fact about exponential families.

2.3. **Online Variational Inference.** The update equations for batch mean field variational inference for LDA present a clear inefficiency. We must loop over all of the documents in the corpus and compute the variational parameters γ and ϕ before we can update λ even once. This process is costly if the document collection is large, thus motivating the development of an online inference algorithm for LDA. In practice, we randomly initialize λ when we implement the algorithm. Note that the updates for γ and ϕ depend on the choice of λ though, so during early iterations of the algorithm we use values of λ far from their optimal values to compute all the document specific ϕ and γ. However, if we are able to learn something about λ from only a subsample of the data, it may make more sense to re-update λ first instead of continuing to optimize every γ and ϕ [3].

The solution to the points just raised is to use stochastic optimization. Whereas batch variational inference relies on the exact gradient of the ELBO, in stochastic or online variational inference, we use a noisy calculation of the gradient of the objective function, obtained by subsampling the data. Instead of using the information from the entire corpus to re-update λ, we instead take smaller subsamples of the whole corpus, and use those to noisily update λ (and compute a noisy estimate of the ELBO). If the expectation of the noisy gradient is equal to the gradient and if the step size of each iteration decreasing according to a specific schedule, then convergence of the stochastic algorithm to a local optima is guaranteed [10].

We may also premultiply the gradient by a particular positive semidefinite matrix (the inverse of the Fisher information matrix), transforming the Euclidean gradient into the natural gradient [3,6]. Whereas the Euclidean gradient gives the direction of steepest ascent in Euclidean space with respect to the Euclidean distance metric, the natural gradient gives the direction of steepest ascent in the general Riemannian space where distance is defined by a symmetric version of KL Divergence. The natural gradient points in a more informative direction than the Euclidean gradient, and its use leads to better convergence of the algorithm.

The main structure of the online learning algorithm is as follows. First, the step size schedule is set accordingly, and λ is initialized randomly. Then, during each iteration of the algorithm, we first run a local step, subsampling some small batch of documents. We initialize γ randomly for this batch, then iteratively update γ and ϕ for the mini-batch via Equations 6,7 until γ converges. Finally, we update the λ via the revised updates, where n_{ts} is the sth document in mini-batch t:

$$\tilde{\lambda_{kw}} = \eta + \frac{D}{S} \sum_{s} n_{tsw} \phi_{tskw} \tag{11}$$

$$\lambda = (1 - \rho_t)\lambda + \rho_t \tilde{\lambda} \tag{12}$$

Equation 11 describes how to incorporate knowledge of the global variables given the information from our mini-batch, and Equation 12 describes how heavily to weight this new value of λ from the previous one. Note that ρ_t is our step size, which we initialize as $\rho_t = (\tau_0 + t)^{-\kappa}$; for convergence we require that $\kappa \in (0.5, 1]$ and $\tau_0 \geq 0$ so that $\sum \rho_t = \infty$ and $\sum \rho_t^2 < \infty$ [11]. Finally, we continue

this procedure indefinitely, until a noisy estimate of the ELBO that we recalculate after all the updates converges.

In summary, the online or stochastic variational inference algorithm that we implement and use can be interpreted as stochastic natural gradient descent of the variational objective. It improves on the naive batch variational inference through its use of the natural gradient and its use of stochastic optimization to allow it to scale to massive data.

2.4. **Quantitative Evaluation Methods.** Given the unsupervised nature of topic modeling, it can be particularly hard to quantitatively evaluate their performance. Unlike in instances of supervised learning, where there is a true label to our data, when applying topic models to real data, there is no true label. LDA and other topic models are latent variable models intended to model and discover hidden structure in the data, even when we lack a labeling of any kind. This makes the task of evaluating topic models especially difficult. One application driven approach is to test performance on a specific task, like document classification or information retrieval [7].

An alternate method frequently used is based on the idea of estimating the probability of held out documents given training documents. One common and simple quantitative metric for calculating this is called perplexity. Defined to be the inverse of the geometric mean per-word likelihood, perplexity intuitively measures the uncertainty in predicting a single word [4]. Typically we calculate it first by training the topic model and learning the optimal settings for parameters and latent variables, and then use a held-out test set to evaluate the perplexity of words in held-out documents. Several other quantitative techniques for evaluating topic models are explored in [7]. Mathematically:

$$perplexity(D_{test}) = \exp\{-\frac{\sum_d \log\, p(w_d)}{\sum_d N_d}\} \tag{13}$$

where we sum over the documents in the test set, and N_d is the number of words in document d. Other methods also exist for calculating the probability of held out documents given training documents, including importance sampling, annealed importance sampling, Chib-style estimation, and the Harmonic mean method [7].

3. Data, Preprocessing, and Implementations

3.1. **Synthetic Data.** We first analyzed two synthetic data sets. The synthetic data is generated from simple bars topics, using the generative model described by LDA. We generate synthetic bars data as in [4]. In this setting, we create a synthetic dataset of D images, where each image is analogous to a document in a corpus. Each pixel corresponds to a unique word in the fixed vocabulary, and the intensity of that pixel is the frequency of the word in the document. In this framework, topics are obvious patterns of pixels. Specifically, we create simple bars topics, where a single row or column of pixels are active, or used, with the rest inactive. See Figure 2 for a simple example of the bars topics used to generate fake data, for the case of four topics. Note that we restrict our images to be square, so that for K topics the size of our vocabulary is K^2 unique pixels or words.

We ran experiments on synthetic data sets of two sizes. In the first small example, we created 250 images, each with 50 "words", from the four bars topics in Figure 2. This was done simply by applying the generative process of LDA to produce images that exhibit these four topics in varying proportions. Figure 3 shows a subset of the data used in this experiment. Note as in [4] that we set $\alpha = 1$ so that while in some documents it is obvious which topics generated it, in general the images are noisy and don't immediately reveal which topics were most active for them.

Our second synthetic dataset was identical except much larger. Figure 4 shows 24 documents from it. In this case the full data set contained 100,000 documents, drawn from 35 topics (so a unique fixed vocabulary of 1225 pixels/words), with 1000 words per document.

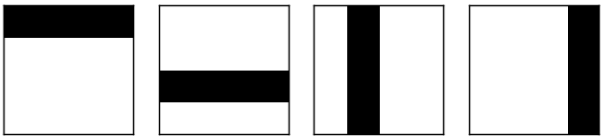

FIGURE 2. Synthetic Data Bars Topics, K=4

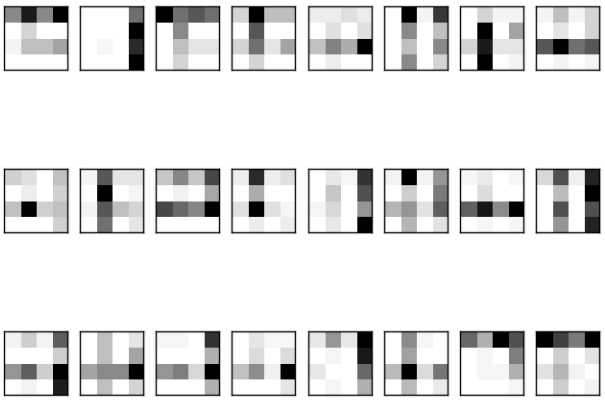

FIGURE 3. 24 Synthetic Documents, W=50, K=4

3.2. **Real Data.** Our primary real data set was a subset of Project Gutenberg, an online repository of literary work freely available. It consists of more than 42,000 electronic works. We took 17,000 of these, and split them into 225,900 documents of 2,000 words or fewer each, throwing out documents with fewer than 100 words. This dramatically increases the size of our training set, as each original document contains many more words than is necessary and would have yielded poor results. Note that we set aside a random subset of 1,000 documents in the total data as a testing set for model selection, and trained on the remaining 224,900 documents. In the corpus there are 1.5 million unique words in total that appear 440 million times, excluding a list of standard stop words (non-informative words like "and", "the", single characters, etc.). We prune this vocabulary down to 15,000 words, via tf-idf (term frequency - inverse document frequency), a popular scheme for word

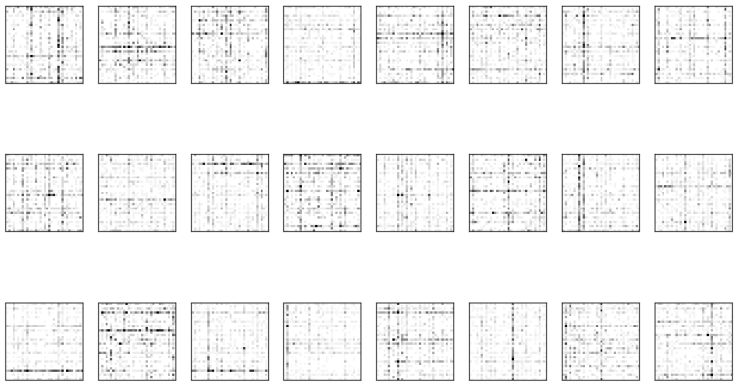

FIGURE 4. 24 Synthetic Documents, W=1000, K=35

pruning, by assigning each unique term a simple score capturing word importance, and retaining only the top 1 percent of words in this case [12].

The Shakespeare corpus is a small subset of the larger Gutenberg corpus. It was originally 44 distinct works, which we split into 840 documents of 500 words or fewer each. There were 27,000 unique terms that appeared 411,000 times, which we pruned to 5,400 words by retaining the top 20 percent via tf-idf and removing stop words again.

3.3. **Implementations.** A significant amount of code for this project is unique and written by us. We wrote unique code to parse the raw text into word counts, and also wrote code to preprocess the data and remove the majority of uninteresting words, as described previously. We also wrote the code that stores our data in text files that are loaded in the main LDA implementation. Additionally, all of the experiment code is our own, including the experiment code for the synthetic bars, for Shakespeare, and for calculating perplexity in our experiments on the Gutenberg corpus. Additionally we wrote code to parallelize our experiments for use on a cluster.

Our actual implementations of online LDA and Batch LDA draws heavily from code written by Matthew Hoffman from [6], although we derived and worked through everything he does in his code, line by line and wrote it to fit our unique dataset. In our implementation, we utilize several numerical tricks that Hoffman uses in order to have code that runs efficiently enough to handle data the size of Gutenberg in a feasible amount of time. In particular, we utilize his trick of implicitly calculating the ϕ variational parameters, a trick that saves on time and space. We also make use of the well-known log-sum-exp trick at one point following his idea, to avoid numerical underflow in one of the calculations. To save on memory, the only values we ever store explicitly are the global topic parameters λ, so after running the algorithms we will need to run a last E-step, using the optimal λ, if we wish to save the values of γ for interesting documents. We do this later on using our own code, to produce output yielding the topic proportions for interesting documents. Finally, although Hoffman's code was a starting point, we had to heavily modify parts of his implementation to handle our specific dataset. His original code was intended to be used in a streaming setting, whereas we have a fixed, large dataset, so we had to rethink an efficient way to run the learning algorithm without loading all of our data into memory. Matthew Hoffman's code can be found for reference in the folder we turned in labeled "BLEIonlinelda," along with some other external software that we used for reference. Implementations were done in Numpy.

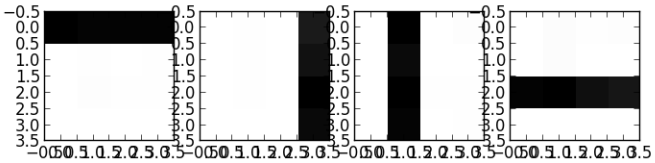

FIGURE 5. Learned topics, batch

4. RESULTS

4.1. **Synthetic Data Results.** From our first synthetic dataset, we obtained several interesting results. The first was that both batch and online were able to learn our topics from the generated synthetic documents, which proves our implementations work and are free of errors. On the small experiment, (250 documents, 50 words, 4 topics) batch variational inference converged quickly (46 seconds, 79 iterations). We ran the online version at various batch sizes of 125, 50, and 25 which took 258, 260, and 131 min. and 39,000, 100,000, and 90,000 iterations, respectively. However, our convergence test was solely for the ELBO to converge within a tiny tolerance. We did not check earlier on to see if the online versions learned meaningful topics early on but simply bounced around local optima for a long time until convergence, as the step sizes slowly decreased. It makes sense that batch, which always takes an exact step in the right direction, converges much quicker for this example, since it is relatively cheap for it to compute an exact gradient step. Note in Figures 5,6 the learned bars from batch and the 25 mini-batch size for online (50 and 125 were similar).

For the larger experiment (100,000 documents, 1000 words per document, 35 topics) the batch inference algorithm could not finish a single iteration after 6 hours, while the online version, with a batch size of 1000, only finished 250. While this was not close to convergence, it does indicate some learning, and shows that online is more efficient for large data. However, we did not want to run it for much longer on our personal laptops. For our larger experiments later on Gutenberg, we used a 4-core, 8-GB RAM desktop computer, since the Anthill computing cluster was out of operation for unexpected maintenance. Although not as long as later experiments, this experiment on a large set of synthetic data is still useful in that it shows online clearly outperforms batch for large data, on the same rough scale (though still a bit smaller) than Gutenberg.

4.2. **Shakespeare Results.** We first ran both Batch and Online on our Shakespeare corpus, which was able to generate some interesting topics. See Table 1 below for a few examples. Although the results were not great and are rather unsurprising, they make sense and are a good indication that our implementation is working properly. First, the typical list of stop word didn't include common antiquated words in Shakespeare's works like thy, thee, and thou, etc, and several topics consisted entirely of these words. Also, a number of topics were just the names of all of the characters from a single play. While these topics are not interesting, they make sense since at the beginning of each

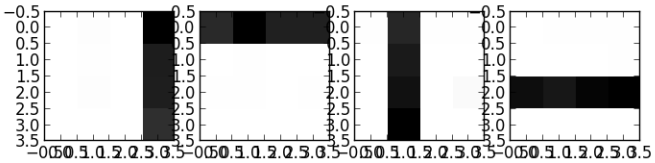

FIGURE 6. Learned topics, online (S=25)

line in a play is the character name, so the character names are bound to be the most strongly co-occurring words in the entire corpus. Also, note the funny topic about Project Gutenberg; this arises because the beginning of every raw text file is a header with info about Project Gutenberg. This was not an issue on the full Gutenberg corpus that we ran, as there are many more works in that to balance out some of these issues.

eyes	project	love
dead	Gutenberg	Doth
hubert	Tm	Fair
death	Works	Hath
blood	Lord	Heart
sorrow	King	Eye
doth	Electronic	Eyes
grief	Keeper	Mine
Tears	Sir	Night
hand	ye	true

TABLE 1. Top 10 words for three Shakespeare topics

4.3. **Gutenberg Results.** Running LDA on the Gutenberg Corpus required the use of the online inference algorithm to be able to get results within a feasible amount of time. We ran each of our experiments of LDA for 48 hours on a 4-core, 8-GB-RAM desktop computer. Although we had originally hoped to run our data on the anthill cluster for a longer period of time, the cluster was down for a long period of maintenance that forced us to alter our plans. Although we would have achieved identical results on anthill, it likely would have run much faster as each job would have had its own node. Instead, using a desktop with only 4 cores for 12 jobs running in parallel caused everything to run slower than it should have on a true cluster.

Our results include both quantitative and qualitative measures, due to the nature of the project. Our quantitative analysis tested the performance of different models. We controlled to test the performance using different numbers of fixed topics, K, and to test the performance using different batch sizes, S. As described previously, we used perplexity, defined to be the inverse of the geometric mean per-word likelihood, to measure the fit of each model. Recall that lower perplexity scores are better.

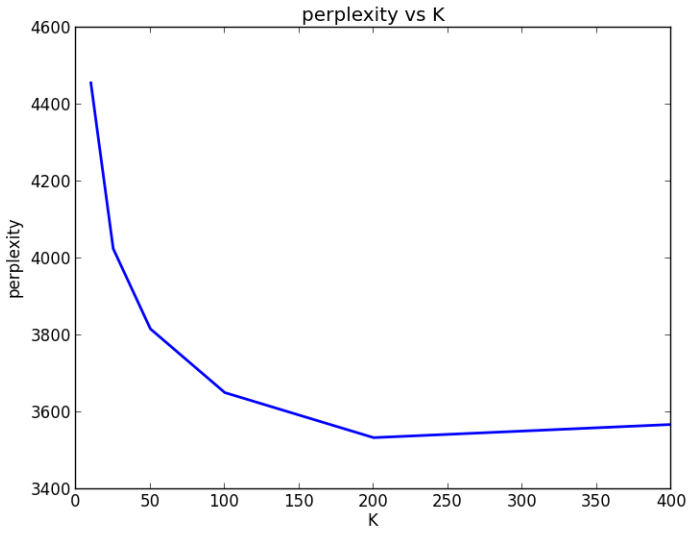

FIGURE 7. Perplexity vs K (S=4096)

We first looked at how K, the number of topics (which is fixed a priori), affected the perplexity of the model on the test data. To perform this analysis, we held all other variables fixed to the optimal or default values in [6] and ran the algorithm using six different values of K, ranging from 10 to 400. We kept the batch size, S, equal to 4096 for each of these runs, as it was found to be the optimal value of S in [6]. Our results showed that $K = 200$ was the optimum value, giving the lowest perplexity. Further, our analysis exhibited the common U-shaped curve in model selection, in Figure 7. The overly complex model (largest K) and overly simple model (smallest K) were outperformed by an appropriate medium choice for K, since they tend to overfit and underfit, respectively.

We also varied batch size, S, over six values ranging from 16 to 16384, keeping all other values held constant. We held the number of topics as $K = 100$ for these runs, as it was found to be the optimal value of K by [6], even though we ultimately found slightly better performance with $K = 200$. The results of these runs showed a similar trend to the results of varying K. Smaller and larger values of S were found to give less good fits to the data, as demonstrated by the standard U-shaped curve seen in Figure 8. The optimal value, according to our results was found to be $S = 1024$, a different result from [6]. However, ideally our experiments would have been run on a cluster where they would have more resources and more time to run. Not all of our experiments were able to be run to convergence due to time constraints and the inability to use anthill, so we had to truncate our jobs at 48 hours. Some of the plots might have looked different if run in better circumstances.

Finally, we looked at perplexity vs. the logarithm of the number of iterations run (the logarithm simply to bring values that vary exponentially from 16 to 16384 on a more uniform scale). This plot is interesting and gives some sense for how quickly the different runs with varying mini-batch sizes converged. Ideally the independent variable in the plots would be time and not iteration number, but since our jobs were run simultaneously on a desktop with limited resources, much of the time might have been due to computational limits and not the true speed of the algorithms. As expected, smaller batch sizes take more iterations to converge, while larger batches converge in fewer iterations (although each iteration takes considerably longer).

4.3.1. *Qualitative Results.* For our qualitative analysis, we chose to look exclusively at the topics found using the values $K = 200$ and $S = 4096$, as this was the run resulting in the lowest perplexity

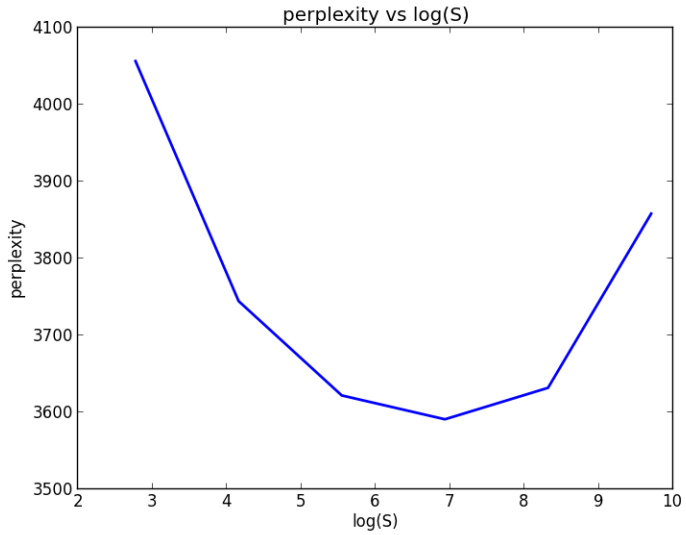

FIGURE 8. Perplexity vs log(S) (K=100)

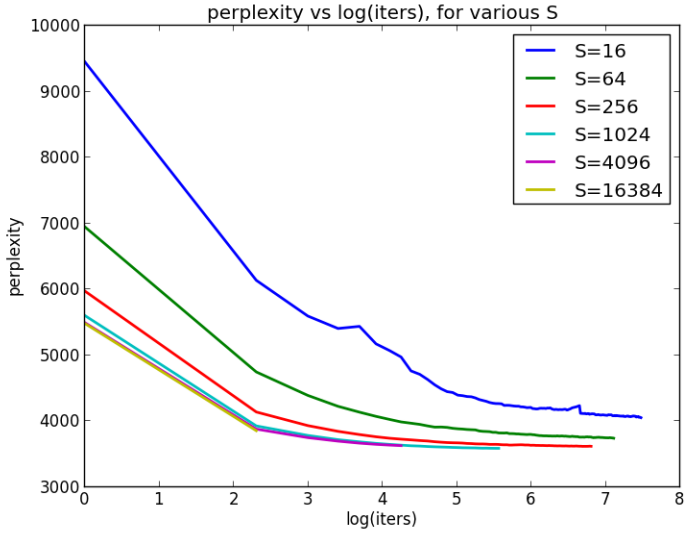

FIGURE 9. Perplexity vs log(tier), various S

of all our runs. The topics that we found revealed that LDA did work on our data, but the topics were varied with respective to how informative they were of the works in the corpus. Further, many of the topics seemed to be strongly related. In particular, topics about god and religion, as well as topics about war, were prevalent. However, within these categories, the topics did show subtle, yet important differences regarding their context. For example, though each topic in Table 2 clearly relates to god, religions and Christianity, the topic in column 4 is clearly demonstrative of the more academic and historical work related to religion. Similarly, though all of the topics in Table 3 relate to war and battles, the topic in the second column is clearly related to the civil war in particular. It is possible that perhaps the model with $K = 200$ was not actually the best model,

even though the perplexity scores indicated it was; a smaller value of K may have combined some of these redundant topics together.

heart	god	god	god
love	lord	heart	religion
god	son	love	history
father	israel	jesus	religious
ming	things	things	christian
poor	jesus	say	greek
say	earth	Christ	ancient
hope	heaven	soul	gods
soul	king	faith	christianity
death	children	think	form

TABLE 2. Top 10 words for four topics relating to God, Religion and Christianity found in Gutenberg, K=200, S=4096

enemy	general	major
general	army	officer
troops	grant	soldiers
army	colonel	officers
force	war	soldiers
battle	president	colonel
attack	confederate	captain
line	sherman	sergeant
war	railroad	left
guns	union	regiment

TABLE 3. Top 10 words for three topics relating to war and armies, found in Gutenberg, K=200, S=4096

The prevalence of topics for which the most probable words were character names revealed some of the shortcomings of LDA on literary works. It was expected that many character-name topics would arise given the frequency of character names within most fictional literature. Despite many informative and specific topics found by running LDA on our corpus, a large number of the topics placed high probability on character-names. The top ten words for four of these topics can be seen in Table 4. Clearly these topics do not provide much useful information to a user hoping to utilize the learned topics for a specific information retrieval task. It would have been nice to have observed topics that picked up on actual hidden themes or symbols in fiction (e.g. coming of age, good vs. evil, the American Dream, etc.). However, topic models only pick up on the pure co-occurrences of words, as the algorithm has no knowledge of word meanings, so it is understandable that such desirable topics are not learned.

A final interesting qualitative result is examining the most used topics for hand-selected famous books. To determine these topics, we ran a final E-step of the online inference algorithm, using the final optimal λ values from our experiments (we used $K = 200, S = 4096$), until convergence of the γ variational parameters specific for these sub-documents. Recall that we split up the original text files in the corpus into smaller sub-documents, each with 2000 words or less (but more than 100). Examination of the largest values of γ across the sub-documents that make up one original text file yield interesting results. Tables 5, 6, and 7 show three of the top topics, with top ten words, for the

books *Narrative of the Life of Frederick Douglass*, *The Red Badge of Courage*, and *The Adventures of Tom Sawyer*. The remainder of the 200 topics found on the Gutenberg Corpus, using the parameters $K = 200$ and $S = 4096$, can be found in our code submission. It is recommended that one look at these to understand the quality and variety of topics found.

george	nelly	catherine	elsie
anthony	addison	beauchamp	hilda
georges	pepper	terence	molly
roberts	putnam	cecilia	jessie
adrian	steele	marianne	angela
edgar	nell	isabella	dexter
brian	compound	miss	sophy
dinah	theodora	house	balzac
olivia	ellen	think	melissa
enid	cadets	mrs	lisle

TABLE 4. Top 10 words for four topics that give high probability to character names found in Gutenberg, K=200, S=4096

slavery	heart	dat
slave	love	dey
slaves	god	house
negro	father	white
free	mind	master
negros	poor	say
labor	say	folks
south	hope	ernest
states	soul	wid
property	death	bout

TABLE 5. Top 10 words for the three top topics from *Narrative of the Life of Frederick Douglass*

major	eyes	eyes
officer	hand	face
soldiers	face	looked
officers	head	voice
soldier	saw	room
colonel	turned	hand
captain	feet	moment
sergeant	stood	door
left	moment	night
regiment	looked	thought

TABLE 6. Top 10 words for the three top topics from *The Red Badge of Courage*

tom	mother	eyes
frank	girls	face
ned	boy	looked
boys	school	voice
going	home	room
think	mrs	hand
ill	girl	moment
asked	boys	door
say	children	night
want	mary	thought

TABLE 7. Top 10 words for the three top topics from *The Adventures of Tom Sawyer*

5. Conclusion

Our quantitative analysis showed that for the Gutenberg corpus, the optimal values of S and K were S=1024 and K=200. Note however that a few of the experiments for the largest K and S did not completely converge, due to computational resources. Jobs were truncated at 48 hours.

Our qualitative analysis revealed that LDA worked as expected, discovering interesting latent topics within a large corpus of texts. However, many of the discovered topics were not very informative of the content of the works, and in particular were not thematically meaningful. Many topics consisted primarily of names, presumably found throughout the works. It does not appear that such topics would be of much use in organizing large collections of literature in the same way they have been used for information retrieval tasks in scientific articles.

6. References

(1) D. Blei. Probabilistic Topic Models. *Communications of the ACM*, 55(4):7784, 2012.
(2) http://www.gutenberg.org/wiki/Gutenberg:About
(3) M. Hoffman, D. Blei, J. Paisley, C. Wang. Stochastic Variational Inference. *arXiv 1206.7051*, 2012.
(4) T. Griffiths, M. Steyvers. Finding scientific topics. *PNAS* 101 (Suppl 1): 522835, 2004.
(5) D. Blei, A. Ng, M. Jordan. Latent Dirichlet Allocation. *JMLR*, 2003.
(6) M. Hoffman, D. Blei, F. Bach. Online Learning for Latent Dirichlet Allocation. *NIPS*, 2010.
(7) H. Wallach, I. Murray, R. Salakhutdinov, D. Mimmo. Evaluation Methods for Topic Models. *ICML*, 2009.
(8) B. de Finetti. *Theory of probability. Vol. 1-2.* John Wiley & Sons Ltd., Chichester, 1990. Reprint of the 1975 translation.
(9) Y. Teh, M. Jordan, M. Beal, and D. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 2006. 101[476]:1566-1581.
(10) S. Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2): 251276, 1998.
(11) H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400407, 1951.
(12) G. Salton and M. McGill, editors. Introduction to Modern Information Retrieval. McGraw-Hill, 1983.