MILESTONE 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 as well, 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, it 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. 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 literature, however. For example, while it is straightforward to label a work of nonfiction by its main topics or themes, 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. Our aim is to implement efficiently two different inference algorithms for LDA. We first implement batch mean field variational inference for LDA, first introduced in [5]. However, a problem with this inference algorithm is its inability to scale to massive data. To remedy this, we additionally implement an online learning algorithm for LDA, as described in [6]. This inference algorithm can be interpreted as a stochastic natural gradient descent 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. In Section 3, we discuss the preliminary results we have obtained so far, and highlight what tasks remain and what goals we have accomplished.

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.

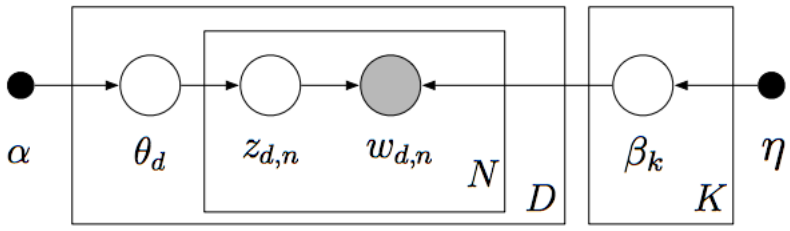

FIGURE 1. LDA Graphical Model

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 variation 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 [SVI].

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. Progress So Far

3.1. **Completed Work.**

3.1.1. *Data.* We use four data sets, two synthetic and two real, to test the algorithms on. The synthetic data is generated from simple fake bars topics, using the generative model described by LDA. The two real data sets come from Project Gutenberg, an online repository of literary work, and a small subset of it. The subset of Gutenberg we investigated was the full works of William Shakespeare.

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 word, and the intensity of that pixel is the frequency of the word in the document. In this setting, topics are obvious patterns of pixels, which in this case is simply a single row or column of pixels that 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.

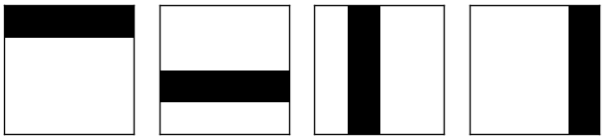

FIGURE 2. Synthetic Data Bars Topics, K=4

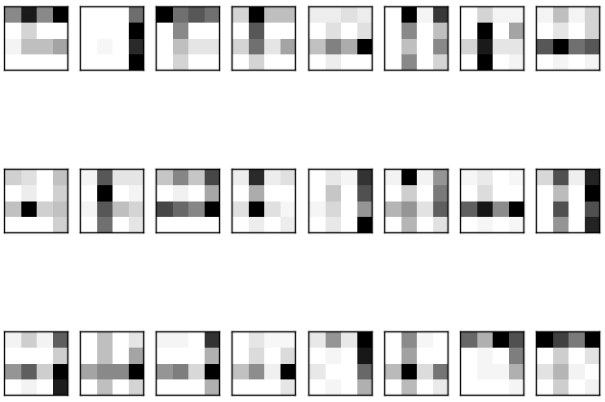

FIGURE 3. 24 Synthetic Documents, W=50, K=4

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.

The first of our real data sets, Project Gutenberg is an online repository of literary work available to the public. It consists of more than 42,000 electronic works. We took 17,000 of these, and split them into close to 100,000 documents of 5,000 words each. This makes LDA more effectively, since in effect we are creating more training examples to train on since each document contains many more words than is necessary to obtain results. In this 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 have pruned this vocabulary down to 15,000 words,

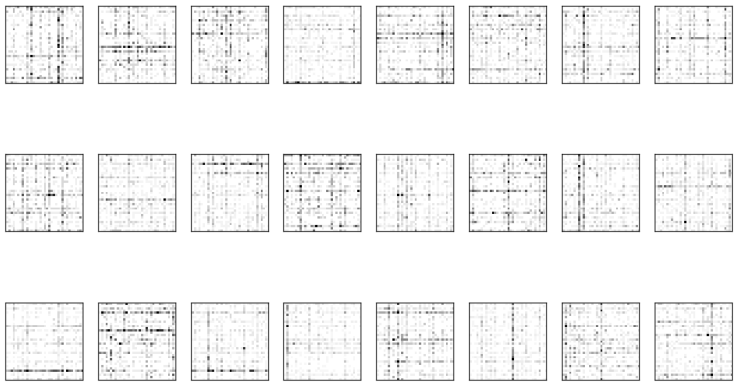

FIGURE 4. 24 Synthetic Documents, W=1000, K=35

via tf-idf (term frequency - inverse document frequency), a popular scheme for word pruning, by assigning each unique term a simple score and retaining only the top 1 percent of words in this case.

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

3.1.2. *Development.* A significant amount of code has been written for our project thus far. 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 meaningful data structures, although we are planning on changing part of this for the final. Additionally, all of the experiment code is our own, and we still have to write more experiment code for Gutenberg as well as the code for whichever quantitative evaluation metrics we decide to implement, like perplexity.

Our actual implementations of online LDA and Batch LDA drew heavily from code written by Matthew Hoffman from [6], although we derived and worked through everything he does in his code, line by line. We utilize several numerical tricks that he 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.

Implementations were done in Numpy. Finally, we are working on a more naive implementation of Batch variational inference for LDA to run as a benchmark against our current implementations. In this version, we write everything on our own, without the use of Hoffman's numerical tricks, and optimize and vectorize our code as much as possible to have a benchmark to compare against (but will use the Hoffman-aided code when running experiments on the full Gutenberg.

3.1.3. *Results.* From our first dataset, we were able to obtain several interesting results. The first was that both batch and online were able to learn our topics from the generated documents, which

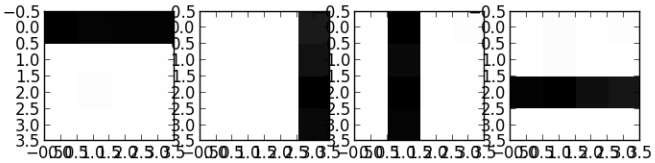

FIGURE 5. Learned topics, batch

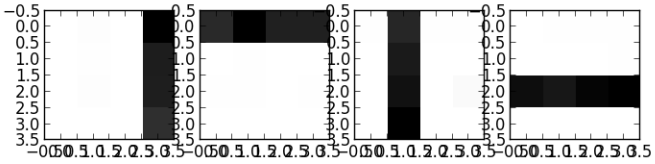

FIGURE 6. Learned topics, online (S=25)

is a good sign that our implementations are accurate and the algorithms work. On the small experiment, (250 documents, 50 words, 4 topics) Batch variational was able to finish very quickly (46 seconds, 79 iterations). Online LDA at various batch sizes of 125, 50, and 25 took 258, 260, and 131 min. and 39,000, 100,000, and 90,000 iterations, respectively. However, the convergence test was solely for the ELBO to converge within a 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. 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) Batch could not finish a single iteration after 6 hours, and Online with a batch size of 1000 only finished 250. While this was not close to convergence, it does indicate some learning. However, we did not want to run it for much longer on our personal laptops. We will not have this problem for running Gutenberg, as we will run our experiments on the Anthill computing cluster. This experiment was 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.

We 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 things are 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 line in a play is the character name, so the character names are bound to be the most important 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. We don't anticipate these problems being an issue on the larger Gutenberg corpus, 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

3.2. **Unfinished Work.** We have yet to run our algorithms on the full Gutenberg corpus. This is due largely to the continued challenge of finding a time and space efficient way of storing and using the massive amount of data that we plan to run through our models. The size of the data is large, but not infeasible to load all at once in RAM. We want to avoid this if possible, however, as this may make our jobs take longer to start on the queue on the cluster if we request large amounts of memory. The size of the data has caused parsing and pruning data to pose a difficulty, and has been a slow process to run locally on our personal laptops, although it is possible. We prune down the number of total and unique words so that our vocabulary is manageable and the total number of words in the dataset is manageable. Our idea for the future is to write the data to text files and only load the relevant portions (for subsamples in online inference) when we need them to avoid loading everything into RAM all at once. We anticipate starting experiments on the full corpus in less than a week.

We also plan to use methods discussed in [7] to evaluate the performance of our models. To do this, we plan to hold some portion of our dataset, and then test the results on the held-out portion of the data. In other words, we approximate the probability of the held-out data given the results of the training data. For better models, this should be higher, on average. This may be implemented

by holding out entire documents, or holding out a portion of words from each document; we will decide which methods to run soon.

Lastly, we must complete the write-up and poster for the final project.

3.3. **Goals Met.** The goals we set for ourselves were high: we had planned to have two variations of LDA coded and to have implemented them using our data set of text files. Though we have verified that our code works, we have not yet been able to run our chosen data set due to its massive size. Thus, we have not fully completed our original goals for the milestone due date, but we have set ourselves up well to complete these goals soon and begin the final stages. We have no worry about completing the final project on time.

4. 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.