Ke Zhai, **Jordan Boyd-Graber**, Nima Asadi, and Mohamad (Jude) Alkhouja. **Mr. LDA: A Flexible Large Scale Topic Modeling Package using Variational Inference in MapReduce**. *ACM International Conference on World Wide Web*, 2012, 10 pages.

```
    @inproceedings{Zhai:Boyd-Graber:Asadi:Alkhouja-2012,
Title = {{Mr. LDA}: A Flexible Large Scale Topic Modeling Package using Variational Inference in MapReduce
Url = {http://umiacs.umd.edu/~jbg//docs/2012_www_mrlda.pdf},
Author = {Ke Zhai and Jordan Boyd-Graber and Nima Asadi and Mohamad Alkhouja},
Booktitle = {ACM International Conference on World Wide Web},
Year = {2012},
Location = {Lyon, France},
}
```

Links:

- Code [http://mrlda.cc]
- Slides [http://umiacs.umd.edu/~jbg/docs/2012_www_slides.pdf]

Contact Jordan Boyd-Graber (jbg@boydgraber.org) for questions about this paper.

Mr. LDA: A Flexible Large Scale Topic Modeling Package using Variational Inference in MapReduce

Ke Zhai
Computer Science
University of Maryland
College Park, MD, USA
zhaike@cs.umd.edu

Jordan Boyd-Graber
iSchool and UMIACS
University of Maryland
College Park, MD, USA
jbg@umiacs.umd.edu

Nima Asadi
Computer Science
University of Maryland
College Park, MD, USA
nima@cs.umd.edu

Mohamad Alkhouja
iSchool
University of Maryland
College Park, MD, USA
khouja@umd.edu

ABSTRACT

Latent Dirichlet Allocation (LDA) is a popular topic modeling technique for exploring document collections. Because of the increasing prevalence of large datasets, there is a need to improve the scalability of inference for LDA. In this paper, we introduce a novel and flexible large scale topic modeling package in MapReduce (Mr. LDA). As opposed to other techniques which use Gibbs sampling, our proposed framework uses variational inference, which easily fits into a distributed environment. More importantly, this variational implementation, unlike highly tuned and specialized implementations based on Gibbs sampling, is easily extensible. We demonstrate two extensions of the models possible with this scalable framework: informed priors to guide topic discovery and extracting topics from a multilingual corpus. We compare the scalability of Mr. LDA against Mahout, an existing large scale topic modeling package. Mr. LDA out-performs Mahout both in execution speed and held-out likelihood.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Clustering—*topic models, scalability, mapreduce*

1. INTRODUCTION

Because data from the web are big and noisy, algorithms that process large document collections cannot solely depend on human annotations. One popular technique for navigating large unannotated document collections is topic modeling, which discovers the themes that permeate a corpus. Topic modeling is exemplified by *Latent Dirichlet Allocation* (LDA), a generative model for document-centric corpora [1]. It is appealing for noisy data because it requires no annotation and discovers, without any supervision, the thematic trends in a corpus. In addition to discovering which topics exist in a corpus, LDA also associates documents with these topics, revealing previously unseen links between documents and trends over time. Although our focus is on text data, LDA is widely used in computer vision [2, 3], computational biology [4, 5], and computational linguistics [6, 7].

WWW 2012, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1229-5/12/04.

In addition to being noisy, data from the web are big. The MapReduce framework for large-scale data processing [8] is simple to learn but flexible enough to be broadly applicable. Designed at Google and open-sourced by Yahoo, Hadoop MapReduce is one of the mainstays of industrial data processing and has also been gaining traction for problems of interest to the academic community such as machine translation [9], language modeling [10], and grammar induction [11].

In this paper, we propose a parallelized LDA algorithm in the MapReduce programming framework (*Mr. LDA*).[1] Mr. LDA relies on variational inference, as opposed to the prevailing trend of using Gibbs sampling. We argue for using variational inference in Section 2. Section 3 describes how variational inference fits naturally into the MapReduce framework. In Section 4, we discuss two specific extensions of LDA to demonstrate the flexibility of the proposed framework. These are an informed prior to guide topic discovery and a new inference technique for discovering topics in multilingual corpora [12]. Next, we evaluate Mr. LDA's ability to scale in Section 5 before concluding with Section 6.

2. SCALING OUT LDA

In practice, probabilistic models work by maximizing the log-likelihood of observed data given the structure of an assumed probabilistic model. Less technically, generative models tell a story of how your data came to be with some pieces of the story missing; inference fills in the missing pieces with the best explanation of the missing variables. Because exact inference is often intractable (as it is for LDA), complex models require approximate inference.

2.1 Why not Gibbs Sampling?

One of the most widely used approximate inference techniques for such models is Markov chain Monte Carlo (MCMC) sampling, where one samples from a Markov chain whose stationary distribution is the posterior of interest [20, 21]. Gibbs sampling, where the Markov chain is defined by the conditional distribution of each latent variable, has found widespread use in Bayesian models [20, 22, 23, 24]. MCMC is a powerful methodology, but it has drawbacks. Convergence of the sampler to its stationary distribution is difficult to diagnose, and sampling algorithms can be slow to converge in high dimensional models [21].

[1]Download the code at `http://mrlda.cc`.

	Framework	Inference	Likelihood Computation	Asymmetric α Prior	Hyperparameter Optimization	Informed β Prior	Multilingual
Mallet [13]	Multi-thread	Gibbs	$\surd$	$\surd$	$\surd$	$\times$	$\surd$
GPU-LDA [14]	GPU	Gibbs & V.B.	$\surd$	$\times$	$\times$	$\times$	$\times$
Async-LDA [15]	Multi-thread	Gibbs	$\surd$	$\times$	$\surd$	$\times$	$\times$
N.C.L. [16]	Master-Slave	V.B.	$\sim$	$\times$	$\times$	$\times$	$\times$
pLDA [17]	MPI & MapReduce	Gibbs	$\sim$	$\times$	$\times$	$\times$	$\times$
Y!LDA [18]	Hadoop	Gibbs	$\surd$	$\surd$	$\surd$	$\times$	$\times$
Mahout [19]	MapReduce	V.B.	$\surd$	$\times$	$\times$	$\times$	$\times$
Mr. LDA	**MapReduce**	**V.B.**	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$

Table 1: Comparison among different approaches. Mr. LDA supports all of these features, as compared to existing distributed or multi-threaded implementations. ($\sim$ - not available from available documentation.)

Blei, Ng, and Jordan presented the first approximate inference technique for LDA based on variational methods [1], but the collapsed Gibbs sampler proposed by Griffiths and Steyvers [23] has been more popular in the community because it is easier to implement. However, such methods inevitably have intrinsic problems that lead to difficulties in moving to web-scale: shared state, randomness, too many short iterations, and lack of flexibility.

Shared State.

Unless the probabilistic model allows for discrete segments to be statistically independent of each other, it is difficult to conduct inference in parallel. However, we want models that allow specialization to be shared across many different corpora and documents when necessary, so we typically cannot assume this independence.

At the risk of oversimplifying, collapsed Gibbs sampling for LDA is essentially multiplying the number of occurrences of a topic in a document by the number of times a word type appears in a topic across all documents. The former is a document-specific count, but the latter is shared across the entire corpus. For techniques that scale out collapsed Gibbs sampling for LDA, the major challenge is keeping these second counts for collapsed Gibbs sampling consistent when there is not a shared memory environment.

Newman et al. [25] consider a variety of methods to achieve consistent counts: creating hierarchical models to view each slice as independent or simply syncing counts in a batch update. Yan et al. [14] first cleverly partition the data using integer programming (an NP-Hard problem). Wang et al. [17] use message passing to ensure that different slices maintain consistent counts. Smola and Narayanamurthy [18] use a distributed memory system to achieve consistent counts in LDA, and Ahmed et al. [26] extend the approach more generally to latent variable models.

Gibbs sampling approaches to scaling thus face a difficult dilemma: completely synchronize counts, which can compromise scaling, or allow for inconsistent counts, which could negatively impact the quality of inference. In contrast to some engineering work-arounds, variational inference provides a *mathematical* solution of how to scale inference for LDA. By assuming a variational distribution that treats documents as independent, we can parallelize inference without a need for synchronizing counts (as required in collapsed Gibbs sampling).

Randomness.

By definition, Monte Carlo algorithms depend on randomness. However, MapReduce implementations assume that every step of computation will be the same, no matter where or when it is run. This allows MapReduce to have greater fault-tolerance, running multiple copies of computation subcomponents in case one fails or takes too long. This is, of course, easily fixed (e.g. by seeding a random number generator in a shard-dependent way), but it adds another layer of complication to the algorithm. Variational inference, given an initialization, is deterministic, which is more in line with MapReduce's system for ensuring fault tolerance.

Many Short Iterations.

A single iteration of Gibbs sampling for LDA with K topics is very quick. For each word, the algorithm performs a simple multiplication to build a sampling distribution of length K, samples from that distribution, and updates an integer vector. In contrast, each iteration of variational inference is difficult; it requires the evaluation of complicated functions that are not simple arithmetic operations directly implemented in an ALU (these are described in Section 3).

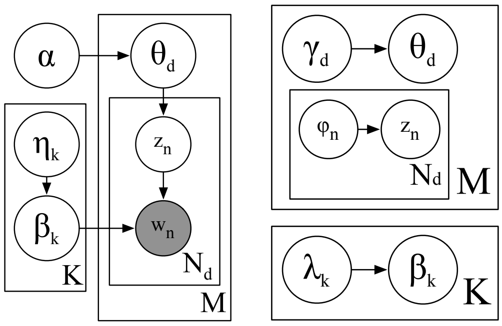
This does not mean that variational inference is slower, however. Variational inference typically requires dozens of iterations to converge, while Gibbs sampling requires thousands (determining convergence is often more difficult for Gibbs sampling). Moreover, the requirement of Gibbs sampling to keep a consistent state means that there are many more synchronizations required to complete inference, increasing the complexity of the implementation and the communication overhead. In contrast, variational inference requires synchronization only once per iteration (dozens of times for a typical corpus); in a naïve Gibbs sampling implementation, inference requires synchronization after every word in every iteration (potentially billions of times for a moderately-sized corpus).

Extension and Flexibility.

Compared to Mr. LDA, many Gibbs samplers are highly tuned specifically for LDA, which restricts extensions and enhancements, one of the key benefits of the statistical approach. The techniques to improve inference for collapsed Gibbs samplers [27] typically reduce flexibility; the factorization of the conditional distribution is limited to LDA's explicit formulation. Adapting such tricks beyond LDA requires repeating the analysis to refactorize the conditional distribution. In Section 4.1 we add an informed prior to topics' word distribution, which guides the topics discovered by the framework to psychologically plausible concepts. In Section 4.2, we adapt Mr. LDA to learn multilingual topics.

2.2 Variational Inference

An alternative to MCMC is variational inference. Variational methods, based on techniques from statistical physics, use optimization to find a distribution over the latent variables that is close to the posterior of interest [28, 29]. Variational methods provide effective approximations in topic models and nonparametric Bayesian models [30, 31, 32]. We believe that it is well-suited to MapReduce.



(a) LDA (b) Variational

Figure 1: Graphical model of LDA and the mean field variational distribution. Each latent variable, observed datum, and parameter is a node. Lines between represent possible statistical dependence. Shaded nodes are observations; rectangular plates denote replication; and numbers in the bottom right of a plate show how many times plates' contents repeat. In the variational distribution (right), the latent variables θ, β, and z are explained by a simpler, fully factorized distribution with variational parameters γ, λ, and ϕ. The lack of inter-document dependencies in the variational distribution allows the parallelization of inference in MapReduce.

Variational methods enjoy clear convergence criterion, tend to be faster than MCMC in high-dimensional problems, and provide particular advantages over sampling when latent variable pairs are not conjugate. Gibbs sampling requires conjugacy, and other forms of sampling that can handle non-conjugacy, such as Metropolis-Hastings, are much slower than variational methods.

With a variational method, we begin by positing a family of distributions $q \in Q$ over the same latent variables Z with a simpler dependency pattern than p, parameterized by Θ. This simpler distribution is called the variational distribution and is parameterized by Ω, a set of variational parameters. With this variational family in hand, we optimize the *evidence lower bound* (ELBO),

$$\mathcal{L} = \mathbb{E}_q \left[\log \left(p(\mathbf{D}|Z)p(Z|\Theta)\right)\right] - \mathbb{E}_q \left[\log q(Z)\right] \qquad (1)$$

a lower bound on the data likelihood. Variational inference fits the variational parameters Ω to tighten this lower bound and thus minimizes the Kullback-Leibler divergence between the variational distribution and the posterior.

The variational distribution is typically chosen by removing probabilistic dependencies from the true distribution. This makes inference tractable and also induces independence in the variational distribution between latent variables. This independence can be engineered to allow parallelization of independent components across multiple computers.

Maximizing the global parameters in MapReduce can be handled in a manner analogous to EM [33]; the expected counts (of the variational distribution) generated in many parallel jobs are efficiently aggregated and used to recompute the top-level parameters.

2.3 Related Work

Nallapati, Cohen and Lafferty [16] extended variational inference for LDA to a parallelized setting. Their implementation uses a master-slave paradigm in a distributed environment, where all the slaves are responsible for the E-step and the master node gathers all the intermediate outputs from the slaves and performs the M-step. While this approach parallelizes the process to a small-scale distributed environment, the final aggregation/merging showed an I/O bottleneck that prevented scaling beyond a handful of slaves because the master has to explicitly read all intermediate results from slaves.

Mr. LDA addresses these problems by parallelizing the work done by a single master (a reducer is only responsible for a single topic) and relying on the MapReduce framework, which can efficiently marshal communication between compute nodes. Building on the MapReduce framework also provides advantages for reliability and monitoring not available in an *ad hoc* parallelization framework.

The MapReduce [8] framework was originally inspired from the map and reduce functions commonly used in functional programming. It adopts a divide-and-conquer approach. Each *mapper* processes a small subset of data and passes the intermediate results as key value pairs to *reducers*. The reducers receive these inputs in sorted order, aggregate them, and produce the final result. In addition to mappers and reducers, the MapReduce framework allows for the definition of *combiners* and *partitioners*. Combiners perform local aggregation on the key value pairs after map function. Combiners help reduce the size of intermediate data transferred and are widely used to optimize a MapReduce process. Partitioners control how messages are routed to reducers.

Mahout [19], an open-source machine learning package, provides a MapReduce implementation of variational inference LDA, but it lacks features required by mature LDA implementations such as supplying per-document topic distributions and optimizing hyperparameters (for an explanation of why this is essential for model quality, see Wallach et al.'s "Why Priors Matter" [34]). Without per-document topic distributions, many of the downstream applications of LDA (e.g. document clustering) become more difficult.

Table 1 provides a general overview and comparison of features among different approaches for scaling LDA. Mr. LDA is the only implementation which supports all listed capabilities in a distributed environment.

3. MR. LDA

LDA assumes the following generative process to create a corpus of M documents with N_d words in document d using K topics.

1. For each topic index $k \in \{1, \ldots K\}$, draw topic distribution $\boldsymbol{\beta}_k \sim \text{Dir}(\boldsymbol{\eta}_k)$
2. For each document $d \in \{1, \ldots M\}$:
   (a) Draw document's topic distribution $\boldsymbol{\theta}_d \sim \text{Dir}(\boldsymbol{\alpha})$
   (b) For each word $n \in \{1, \ldots N_d\}$:
      i. Choose topic assignment $z_{d,n} \sim \text{Mult}(\boldsymbol{\theta}_d)$
      ii. Choose word $w_{d,n} \sim \text{Mult}(\boldsymbol{\beta}_{z_{d,n}})$

In this process, Dir() represents a Dirichlet distribution, and Mult() is a multinomial distribution. $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are parameters.

The mean-field variational distribution q for LDA breaks the connection between words and documents

$$q(z, \theta, \beta) = \prod_k \text{Dir}(\beta_k \,|\, \lambda_k) \prod_d \text{Dir}(\theta_d \,|\, \gamma_d)\text{Mult}(z_{d,n} \,|\, \phi_{d,n}),$$

and when used in Equation 1 yields updates that optimize $\mathcal{L}$, the lower bound on the likelihood. In the sequel, we take these updates as given, but interested readers can refer to the appendix of Blei et al. [1]. Variational EM alternates between updating the expectations of the variational distribution q and maximizing the probability of the parameters given the "observed" expected counts.

The remainder of the paper focuses on adapting these updates into the MapReduce framework and challenges of working at a large scale. We focus on the primary components of a MapReduce

Algorithm 1 Mapper

Input:
KEY - document ID $d \in [1, C]$, where $C = |\mathcal{C}|$.
VALUE - document content.

Configure

Load in α's, λ's and γ's from *distributed cache*.
Normalize λ's for every topic.

Map

Initialize a zero $V \times K$-dimensional matrix ϕ.
Initialize a zero K-dimensional row vector σ.
Read in document content $\|w_1, w_2, \ldots, w_V\|$
repeat
  for all $v \in [1, V]$ **do**
    for all $k \in [1, K]$ **do**
      Update $\phi_{v,k} = \frac{\lambda_{v,k}}{\sum_v \lambda_{v,k}} \cdot \exp \Psi\left(\gamma_{d,k}\right)$.
    end for
    Normalize ϕ_v, set $\sigma = \sigma + w_v \phi_{v,*}$
  end for
  Update row vector $\gamma_{d,*} = \alpha + \sigma$.
until convergence
for all $k \in [1, K]$ **do**
  for all $v \in [1, V]$ **do**
    Emit $\langle k, v \rangle : w_v \phi_{v,k}$.
  end for
  Emit $\langle \triangle, k \rangle : \left(\Psi\left(\gamma_{d,k}\right) - \Psi\left(\sum_{l=1}^{K} \gamma_{d,l}\right)\right)$. {Section 3.4}
  Emit $\langle k, d \rangle : \gamma_{d,k}$ to file.
end for
Aggregate $\mathcal{L}$ to global counter. {ELBO, Section 3.5}

Algorithm 2 Reducer

Input:
KEY - key pair $\langle p_{\mathsf{left}}, p_{\mathsf{right}} \rangle$.
VALUE - an iterator $\mathcal{I}$ over sequence of values.

Reduce

Compute the sum σ over all values in the sequence $\mathcal{I}$. σ is un-normalized λ if $p_{\mathsf{left}} \neq \triangle$ and α sufficient statistics (refer to Section 3.4 for more details) otherwise.
Emit $\langle p_{\mathsf{left}}, p_{\mathsf{right}} \rangle : \sigma$.

algorithm: the mapper, which processes a single unit of data (in this case, a document); the reducer, which processes a single view of globally shared data (in this case, a topic parameter); the partitioner, which distributes the workload to reducers; and the driver, which controls the overall algorithm. The interconnections between the components of Mr. LDA are depicted in Figure 2.

3.1 Mapper: Update ϕ and γ

Each document has associated variational parameters γ and ϕ. The mapper computes the updates for these variational parameters and uses them to create the sufficient statistics needed to update the global parameters. In this section, we describe the computation of these variational updates and how they are transmitted to the reducers.

Given a document, the updates for ϕ and γ are

$$\phi_{v,k} \propto \mathbb{E}_q\left[\beta_{v,k}\right] \cdot e^{\Psi(\gamma_k)}, \qquad \gamma_k = \alpha_k + \sum_{v=1}^{V} \phi_{v,k},$$

where $v \in [1, V]$ is the term index and $k \in [1, K]$ is the topic index. In this case, V is the size of the vocabulary $\mathcal{V}$ and K denotes the total number of topics. The expectation of β under q gives an estimate of how compatible a word is with a topic; words highly compatible with a topic will have a larger expected β and thus higher values of ϕ for that topic.

Algorithm 1 illustrates the detailed procedure of the Map function. In the first iteration, mappers initialize variables, e.g. seed λ with the counts of a single document. For the sake of brevity, we omit that step here; in later iterations, global parameters are stored in *distributed cache* – a synchronized read-only memory that is shared among all mappers [35] – and retrieved prior to mapper execution in a configuration step.

A document is represented as a term frequency sequence $\vec{w} = \|w_1, w_2, \ldots, w_V\|$, where w_i is the corresponding **term frequency in document** d. For ease of notation, we assume the input term frequency vector $\vec{w}$ is associated with all the terms in the vocabulary, i.e., if term t_i does not appear at all in document d, $w_i = 0$.

Because the document variational parameter γ and the word variational parameter ϕ are tightly coupled, we impose a local convergence requirement on γ in the Map function. This means that the mapper alternates between updating γ and ϕ until γ stops changing.

3.2 Partitioner: Evenly Distribute Workloads

The Map function in Algorithm 1 emits sufficient statistics for updating the topic variational distribution λ. These sufficient statistics are keyed by a composite key set $\langle p_{\mathsf{left}}, p_{\mathsf{right}} \rangle$. These keys can take two forms: tuple of topic and word identifier or, when the value represents the sufficient statistics for α updating, a unique value $\triangle$ and a topic identifier.

A partitioner is required to ensure that messages from the mappers are sent to the appropriate reducers. Each reducer is responsible for updating the per-topic variational parameter associated with a single topic indexed by k. This is accomplished by ensuring the partitioner sorts on topic only. A consequence of this is that any reducers beyond the number of topics is superfluous. Given that the vast majority of the work is in the mappers, this is typically not an issue for LDA.

3.3 Reducer: Update λ

The Reduce function updates the variational parameter λ associated with each topic. It requires aggregation over all intermediate ϕ vectors

$$\lambda_{v,k} = \eta_{v,k} + \sum_{d=1}^{C} \left(w_v^{(d)} \phi_{v,k}^{(d)}\right),$$

where $d \in [1, C]$ is the document index and $w_v^{(d)}$ denotes the number of appearances of term v in document d. Similarly, C is the number of documents. Although the variational update for λ does not include a normalization, the expectation $\mathbb{E}_q[\beta]$ requires the λ normalizer. In Mr. LDA, the $\lambda_{v,k}$ parameters are distributed to all mappers, and the normalization is taken care of by the mappers in a configuration step prior to every iteration.

To improve performance, we use combiners to facilitate the aggregation of sufficient statistics in mappers before they are transferred to reducers. This decreases bandwidth and saves the reducer computation.

3.4 Driver: Update α

Effective inference of topic models depends on learning not just the latent variables β, θ, and z but also estimating the hyperparameters, particularly α. The α parameter controls the sparsity of topics in the document distribution and is the primary mechanism that differentiates LDA from previous models like pLSA and LSA; not optimizing α risks learning suboptimal topics [34].

Updating hyperparameters is also important from the perspective of equalizing differences between inference techniques; as long as hyperparameters are optimized, there is little difference between the *output* of inference techniques [36].

The driver program marshals the entire inference process. On the first iteration, the driver is responsible for initializing all the model parameters (K, V, C, η, α); the number of topics K is user specified; C and V, the number of documents and types, is determined by the data; the initial value of α is specified by the user; and λ is randomly initialized or otherwise seeded.

The driver updates α after each MapReduce iteration. We use a Newton-Raphson method which requires the Hessian matrix and the gradient,

$$\alpha_{\mathsf{new}} = \alpha_{\mathsf{old}} - \mathcal{H}^{-1}(\alpha_{\mathsf{old}}) \cdot g(\alpha_{\mathsf{old}}),$$

where the Hessian matrix $\mathcal{H}$ and α gradient are, respectively, as

$$\begin{aligned}
\mathcal{H}(k,l) =& \delta(k,l) C \Psi'(\alpha_k) - C\Psi'\left(\sum_{l=1}^{K} \alpha_l\right), \\
g(k) =& \underbrace{C\left(\Psi\left(\sum_{l=1}^{K}\alpha_l\right) - \Psi(\alpha_k)\right)}_{\text{computed in driver}} \\
&+ \underbrace{\sum_{d=1}^{C} \underbrace{\Psi(\gamma_{d,k}) - \Psi\left(\sum_{l=1}^{K}\gamma_{d,l}\right)}_{\text{computed in mapper}}}_{\text{computed in reducer}}.
\end{aligned}$$

The Hessian matrix $\mathcal{H}$ depends entirely on the vector α, which changes during updating α. The gradient g, on the other hand, can be decomposed into two terms: the *α-tokens* (i.e., $\Psi\left(\sum_{l=1}^{K}\alpha_l\right) - \Psi(\alpha_k)$) and the γ-tokens (i.e., $\sum_{d=1}^{C}\Psi(\gamma_{d,k}) - \Psi\left(\sum_{l=1}^{K}\gamma_{d,l}\right)$). We can remove the dependence on the number of documents in the gradient computation by computing the γ-tokens in mappers. This observation allows us to optimize α in the MapReduce environment.

Because LDA is a dimensionality reduction algorithm, there are typically a small number of topics K even for a large document collection. As a result, we can safely assume the dimensionality of α, $\mathcal{H}$, and g are reasonably low, and additional gains come from the diagonal structure of the Hessian [37]. Hence, the updating of α is efficient and will not create a bottleneck in the driver.

3.5 Likelihood Computation

The driver monitors the ELBO to determine whether inference has converged. If not, it restarts the process with another round of mappers and reducers. To compute the ELBO we expand Equation 1, which gives us

$$\begin{aligned}
\mathcal{L}(\gamma,\phi,\lambda;\alpha,\eta) =& \underbrace{\sum_{d=1}^{C}\Phi(\alpha)}_{\text{driver}} + \underbrace{\sum_{d=1}^{C}(\underbrace{\mathcal{L}_d(\gamma,\phi) + \mathcal{L}_d(\phi) - \Phi(\gamma)}_{\text{computed in mapper}})}_{\text{computed in reducer}} \\
&+ \underbrace{\sum_{k=1}^{K}\Phi(\eta_{*,k})}_{\text{driver / constant}} - \underbrace{\sum_{k=1}^{K}\underbrace{\Phi(\lambda_{*,k})}_{\text{reducer}}}_{\text{driver}}
\end{aligned}$$

where

$$\begin{aligned}
\Phi(\mu) =& \log\Gamma\left(\textstyle\sum_{i=1}\mu_i\right) - \sum_{i=1}\log\Gamma(\mu_i) \\
&+ \sum_i (\mu_i - 1)\left(\Psi(\mu_i) - \Psi\left(\textstyle\sum_j \mu_j\right)\right). \\
\mathcal{L}_d(\gamma,\phi) =& \sum_{k=1}^{K}\sum_{v=1}^{V}\phi_{v,k} w_v \left[\Psi(\gamma_k) - \Psi\left(\textstyle\sum_{i=1}^{K}\gamma_i\right)\right], \\
\mathcal{L}_d(\phi) =& \sum_{v=1}^{V}\sum_{k=1}^{K}\phi_{v,k}\left(\sum_{i=1}^{V} w_i \log\frac{\lambda_{i,k}}{\sum_j \lambda_{j,k}} - \log\phi_{v,k}\right),
\end{aligned}$$

Almost all of the terms that appear in the likelihood term can be computed in mappers; the only term that cannot are the terms that depend on α, which is updated in the driver, and the variational parameter λ, which is shared among all documents. All terms that depend on α can be easily computed in the driver, while the terms that depend on λ can be computed in each reducer.

Thus, computing the total likelihood proceeds as follows: each mapper computes its contribution to the likelihood bound $\mathcal{L}$, and emits a special key that is unique to likelihood bound terms and then aggregated in the reducer; the reducers add topic-specific terms to the likelihood; these final values are then combined with the contribution from α in the driver to compute a final likelihood bound.

3.6 Structural Optimization

In examining Mr. LDA's performance, the two largest performance limitations were the large number of intermediate values being generated by the mappers and the time it takes for mappers to read in the current variational parameters during during the mapper configuration phase.

Reducer Caching.

Recall that reducers sum over ϕ contributions and emit the λ variational parameters, but mappers require a normalized form to compute the expectation with of the topic with respect to the variational distribution. To improve the normalization step, we compute the sum of the λ variational parameters in the reducer [38, 39], and then emit this sum before we emit the other λ terms.

Although this requires $O(V)$ additional memory, it is strictly less than the memory required by mappers, so it in practice improves performance by allowing mappers to more quickly begin processing data.

File Merge.

Loading files in the distributed cache and configuring every mapper and reducer is another bottleneck for this framework. This is especially true if we launch a large number of reducers every iteration — this will result in a large number of small outputs, since

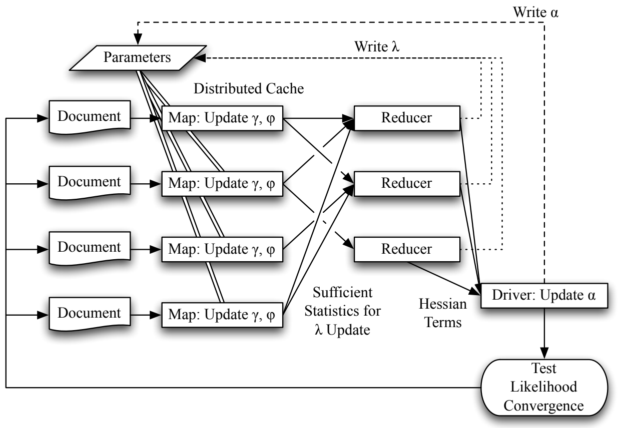


Figure 2: Workflow of Mr. LDA. Each iteration is broken into three stages: computing document-specific variational parameters in parallel mappers, computing topic-specific parameters in parallel reducers, and then updating global parameters in the driver, which also monitors convergence of the algorithm. Data flow is managed by the MapReduce framework: sufficient statistics from the mappers are directed to appropriate reducers, and new parameters computed in reducers are distributed to other computation units via the distributed cache.

Mr. LDA is designed to distribute workload equally. These partial results would waste space if they are significantly smaller than HDFS block size. Moreover, they cause a overhead in file transfer through distributed cache. To alleviate this problem, we merge all relevant output before sending them to distributed cache for the next iteration.

4. FLEXIBILITY OF MR. LDA

In this section, we highlight the flexibility of Mr. LDA to accommodate extensions to LDA. These extensions are possible because of the modular nature of Mr. LDA's design.

4.1 Informed Prior

The standard practice in topic modeling is to use a same symmetric prior (i.e. $\eta_{v,k}$ is the same for all topics k and words v). However, the model and inference presented in Section 3 allows for topics to have different priors. Thus, users can incorporate prior information into the model.

For example, suppose we wanted to discover how different psychological states were expressed in blogs or newspapers. If this were our goal, we might create priors that captured psychological categories to discover how they were expressed in a corpus. The Linguistic Inquiry and Word Count (LIWC) dictionary [40] defines 68 categories encompassing psychological constructs and personal concerns. For example, the *anger* LIWC category includes the words "abuse," "jerk," and "jealous;" the *anxiety* category includes "afraid," "alarm," and "avoid;" and the *negative emotions* category includes "abandon," "maddening," and "sob." Using this dictionary, we built a prior $\boldsymbol{\eta}$ as follows:

$$\eta_{v,k} = \begin{cases} 10, \text{ if } v \in \text{LIWC category}_k \\ 0.01, \text{ otherwise} \end{cases},$$

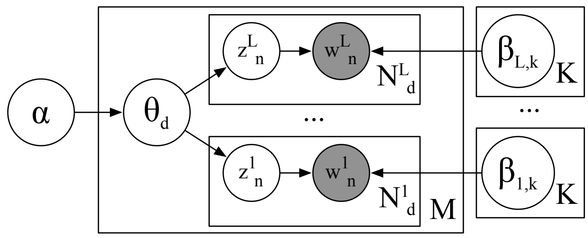


Figure 3: Graphical model for polylingual LDA [12]. Each document has words in multiple languages. Inference learns the topics across languages that have cooccurring words in the corpus. The modular inference of Mr. LDA allows for inference for this model to be accomplished by the same framework created for monolingual LDA.

where $\eta_{v,k}$ is the informed prior for word v of topic k. This is accomplished via a slight modification of the reducer (i.e. to make it aware of the values of η) and leaving the rest of the system unchanged.

4.2 Polylingual LDA

In this section, we demonstrate the flexibility of Mr. LDA by showing how its modular design allows for extending LDA beyond a single language. PolyLDA [12] assumes a document-aligned multilingual corpus. For example, articles in Wikipedia have links to the version of the article in other languages; while the linked documents are ostensibly on the same subject, they are usually not direct translations, and are often written with a culture-specific focus.

PolyLDA assumes that a single document has words in multiple languages, but each document has a common, language agnostic per-document distribution θ (Figure 3). Each topic also has different facets for language; these topics end up being consistent because of the links across language encoded in the consistent themes present in documents.

Because of the modular way in which we implemented inference, we can perform multilingual inference by embellishing each data unit with a language identifier l and change inference as follows:

- Updating λ happens l times, once for each language. The updates for a particular language ignores expected counts of all other languages.
- Updating ϕ happens using only the relevant language for a word.
- Updating γ happens as usual, combining the contributions of all languages relevant for a document.

From an implementation perspective, PolyLDA is a collection of monolingual Mr. LDA computations sequenced appropriately. Mr. LDA's approach of taking relatively simple computation units, allowing them to scale, and preserving simple communication between computation units stands in contrast to the design choices made by approaches using Gibbs sampling.

For example, Smola and Narayanamurthy [18] interleave the topic and document counts during the computation of the conditional distribution using Yao et al.'s "binning" approach [27]. While this improves performance, changing any of the modeling assumptions would potentially break this optimization.

In contrast, Mr. LDA's philosophy allows for easier development of extensions of LDA. While we only discuss two extensions here, other extensions are possible. For example, implementing supervised LDA [41] only requires changing the computation of ϕ and a regression; the rest of the model is unchanged. Implementing syntactic topic models [42] requires changing the mapper to incorporate syntactic dependencies.

5. EXPERIMENTS

We implemented Mr. LDA using Java with Hadoop 0.20.1 and ran all experiments on a cluster containing 16 physical nodes; each node has 16 2.4GHz cores, and has been configured to run a maximum of 6 map and 3 reduce tasks simultaneously. The cluster is usually under a heavy, heterogeneous load. In this section, we document the speed and likelihood comparison of Mr. LDA against Mahout LDA, another large scale topic modeling implementation based on variational inference. We report results on three datasets:

- TREC document collection (disks 4 and 5 [43]), newswire documents from the *Financial Times* and *LA Times*. It contains more than 300k distinct types over half a million documents. We remove types appearing fewer than 20 times, reducing the vocabulary size to approximately 60k.
- The BlogAuthorship corpus [44], which contains about 10 million blog posts from American users. In contrast to the newswire-heavy TREC corpus, the BlogAuthorship corpus is more personal and informal. Again, terms in fewer than 20 documents are excluded, resulting in 53k distinct types.
- Paired English and German Wikipedia articles (more than half a million in each language). As before, we ignore terms appearing in fewer than 20 documents, resulting in 170k English word types and 210k German word types. While each pair of linked documents shares a common subject (e.g. "George Washington"), they are usually *not* direct translations. The document pair mappings were established from Wikipedia's interlingual links.

5.1 Informed Priors

In this experiment, we build the informed priors from LIWC [40] introduced in Section 4.1. We feed the same informed prior to both the TREC dataset and BlogAuthorship corpus. Throughout the experiments, we set the number of topics to 100, with a subset guided by the informed prior.

Table 2 shows topics for both TREC and BlogAuthorship. The prior acts as a seed, causing words used in similar contexts to become part of the topic. This is important for computational social scientists who want to discover how an abstract idea (represented by a set of words) is *actually* expressed in a corpus. For example, public news media (i.e. news articles like TREC) connect positive emotions to entertainment, such as music, film and TV, whereas social media (i.e. blog posts) connect it to religion. The *Anxiety* topic in news relates to middle east, but in blogs it focuses on illness, e.g. bird flu. In both corpora, *Causation* was linked to science and technology.

Using informed priors can discover radically different words. While LIWC is designed for relatively formal writing, it can also discover Internet slang such as "lol" ("laugh out loud") in *Affective Process* category. As a result, an informed prior might be helpful in aligning existing lexical resources with corpora with sparse and/or out-of-dictionary vocabularies, e.g., Twitter data.

On the other hand, some discovered topics do not have a clear relationship with the initial LIWC categories, such as the abbreviations and acronyms in *Discrepancy* category. In other cases, the LIWC categories were different enough from the dataset that model chose not to use topics with ill-fitting priors, e.g. the *Cognitive Process* category.

5.2 Polylingual LDA

As discussed in Section 4.2, Mr. LDA's modular design allows us to consider models beyond vanilla LDA. To the best of our knowledge, we believe this is the first framework for variational inference for polylingual LDA [12], scalable or otherwise. In this experiment, we fit 50 topics to paired English and German Wikipedia articles. We let the program run for 33 iterations with 100 mappers and 50 reducers. Table 3 lists down some words from a set of randomly chosen topics.

The results listed indicates a general equivalent topic layout for both English and German corpus. For example, topic about Europe ("french", "paris", "russian" and "moscow") in English is matched with the topic in German ("frankreich", "paris", "russischen" and "moskau"). Similar behavior was observed for other topics.

The topics discovered by polylingual LDA are not exact matches, however. For example, the second to last column in Table 3is about North America, but the English words focus on Canada, while the corresponding German topic focuses on the United States. Similarly, the forth last column in English contains keywords like "hong", "kong" and "korean", which did not appear in the top 10 words in German. Since this corpus is not a direct translation, these discrepancies might due to a different perspectives, different editorial styles, or different cultural norms.

5.3 Scalability

To measure the scalability and accuracy of Mr. LDA, we compare Mr. LDA with Mahout [19], another large scale topic modeling package based on variational inference. We use Mahout-0.4 as our baseline measure, with a comparable settings in Mr. LDA — we set the memory limit for every mapper and reducer to 2-GB, and start the hyper-parameter α from 1. Mr. LDA continuously updates vector α in the driver program, whereas Mahout does not. All experiments are carried out with 100 mapper instances and 20 reducer instances.

In these set of experiments, we use 90% of the entire TREC corpus as training data, and the rest as our testing data. We then plot the held-out log-likelihood of the test dataset against the training time. Our empirical results show that, with identical data and hardware, Mr. LDA out-performs Mahout LDA.

Both models had identical input and both models ran for 40 iterations. The held-out likelihood was computed using the variational distribution obtained after every iteration. Figure 4 shows the result for 50 topics. Mr. LDA runs faster than Mahout. In addition, Mr. LDA yields a better held-out likelihood than Mahout, probably as a consequence of hyper-parameter updating.

When we double the total number of topics to 100, the difference in processing time is magnified. Mr. LDA converges faster than Mahout, again due to the hyper-parameter updating.

6. CONCLUSION AND FUTURE WORK

Understanding large text collections such as those generated by social media requires algorithms that are unsupervised and scalable. In this paper, we present Mr. LDA, which fulfills both of these requirements. Beyond text, LDA is continually being applied to new fields such as music [45] and source code [46]. All of these domains struggle with the scale of data, and Mr. LDA could help them better cope with large data.

Mr. LDA represents a viable alternative to the existing scalable mechanisms for inference of topic models. Its design easily accommodates other extensions, as we have demonstrated with the

	Affective Processes	Negative Emotions	Positive Emotions	Anxiety	Anger	Sadness	Cognitive Process	Insight	Causation	Discrepancy	Tentative	Certainty
Output from TREC	book	fire	film	al	polic	stock	coalit	un	technolog	pound	hotel	art
	life	hospit	music	arab	drug	cent	elect	bosnia	comput	share	travel	italian
	love	medic	play	israel	arrest	share	polit	serb	research	profit	fish	itali
	like	damag	entertain	palestinian	kill	index	conflict	bosnian	system	dividend	island	artist
	stori	patient	show	isra	prison	rose	anc	herzegovina	electron	group	wine	museum
	man	accid	tv	india	investig	close	think	croatian	scienc	uk	garden	paint
	write	death	calendar	peac	crime	fell	parliament	greek	test	pre	design	exhibit
	read	doctor	movie	islam	attack	profit	poland	yugoslavia	equip	trust	boat	opera
Output from Blog	easili	sorri	lord	bird	iraq	level		god	system	sa	pretty	film
	dare	crappi	prayer	diseas	american	weight		christian	http	ko	davida	actor
	truli	bullshit	pray	shi	countri	disord		church	develop	ang	croydon	robert
	lol	goddamn	merci	infect	militari	moder		jesus	program	pa	crossword	william
	needi	messi	etern	blood	nation	miseri		christ	www	ako	chrono	truli
	jealousi	shitti	truli	snake	unit	lbs		religion	web	en	jigsaw	director
	friendship	bitchi	humbl	anxieti	america	loneli		faith	file	lang	40th	charact
	betray	angri	god	creatur	force	pain		cathol	servic	el	surrey	richard

Table 2: Twelve Topics Discovered from TREC (top) and BlogAuthorship (bottom) collection with LIWC-derived informed prior. The model associates TREC documents containing words like "arab", "israel", "palestinian" and "peace" with *Anxiety*. In the blog corpus, however, the model associates words like "iraq","america*", "militari", "unit", and "force" with the *Anger* category.

English	game	opera	greek	league	said	italian	soviet	french	japanese	album	york	professor
	games	musical	turkish	cup	family	church	political	france	japan	song	canada	berlin
	player	composer	region	club	could	pope	military	paris	australia	released	governor	lied
	players	orchestra	hugarian	played	childern	italy	union	russian	australian	songs	washington	germany
	released	piano	wine	football	death	catholic	russian	la	flag	single	president	von
	comics	works	hungary	games	father	bishop	power	le	zealand	hit	canadian	worked
	characters	symphony	greece	career	wrote	roman	israel	des	korea	top	john	studied
	character	instruments	turkey	game	mother	rome	empire	russia	kong	singer	served	published
	version	composers	ottoman	championship	never	st	republic	moscow	hong	love	house	received
	play	performed	romania	player	day	ii	country	du	korean	chart	county	member
Germany	spiel	musik	ungarn	saison	frau	papst	regierung	paris	japan	album	new	berlin
	spieler	komponist	turkei	gewann	the	rom	republik	franzosischen	japanischen	the	staaten	universitat
	serie	oper	turkischen	spielte	familie	ii	sowjetunion	frankreich	australien	platz	usa	deutschen
	the	komponisten	griechenland	karriere	mutter	kirche	kam	la	japanische	song	vereinigten	professor
	erschien	werke	rumanien	fc	vater	di	krieg	franzosische	flagge	single	york	studierte
	gibt	orchester	ungarischen	spielen	leben	bishof	land	le	jap	lied	washington	leben
	commics	wiener	griechischen	wechselte	starb	italien	bevolkerung	franzosischer	australischen	titel	national	deutscher
	veroffentlic	komposition	istanbul	mannschaft	tod	italienischen	ende	russischen	neuseeland	erreichte	river	wien
	2	klavier	serbien	olympischen	kinder	konig	reich	moskau	tokio	erschien	county	arbeitete
	konnen	wien	osmanischen	platz	tochter	kloster	politischen	jean	sydney	a	gouverneur	erhielt

Table 3: Extracted Polylingual Topics from the Wikipedia Corpus. While topics are generally equivalent (e.g. on "computer games" or "music"), some regional differences are expressed. For example, the "music" topic in German has two words referring to "Vienna" ("wiener" and "wien"), while the corresponding concept in English does not appear until the 15^{th} position.

addition of informed priors and multilingual topic modeling, and the ability of variational inference to support non-conjugate distributions allows for the development of a broader class of models than could be built with Gibbs samplers alone. Mr. LDA, however, would benefit from many of the efficient, scalable data structures that improved other scalable statistical models [47]; incorporating these insights would further improve performance and scalability.

While we focused on LDA, the approaches used here are applicable to many other models. Variational inference is an attractive inference technique for the MapReduce framework, as it allows the selection of a variational distribution that breaks dependencies among variables to enforce consistency with the computational constraints of MapReduce. Developing automatic ways to enforce those computational constraints and then automatically derive inference [48] would allow for a greater variety of statistical models to be learned efficiently in a parallel computing environment.

Variational inference is also attractive for its ability to handle online updates. Mr. LDA could be extended to more efficiently handle online batches in streaming inference [49], allowing for even larger document collections to be quickly analyzed and understood.

7. ACKNOWLEDGMENTS

The authors would also like to thank Dr. Jimmy Lin for valuable comments throughout this project and making data available for the experiments. This research was supported by NSF grant #1018625. Jordan Boyd-Graber is also supported by the Army Research Laboratory through ARL Cooperative Agreement W911NF-09-2-0072. Any opinions, findings, conclusions, or recommendations expressed are the authors' and do not necessarily reflect those of the sponsors.

8. REFERENCES

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

[2] F. Rob, L. Fei-Fei, P. Pietro, and Z. Andrew, "Learning object categories from Google's image search." in *International Conference on Computer Vision*, 2005.

[3] C. Wang, D. Blei, and L. Fei-Fei, "Simultaneous image classification and annotation," in *Computer Vision and Pattern Recognition*, 2009.

[4] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing, "Mixed membership stochastic blockmodels," *Journal of Machine Learning Research*, vol. 9, pp. 1981–2014, 2008.

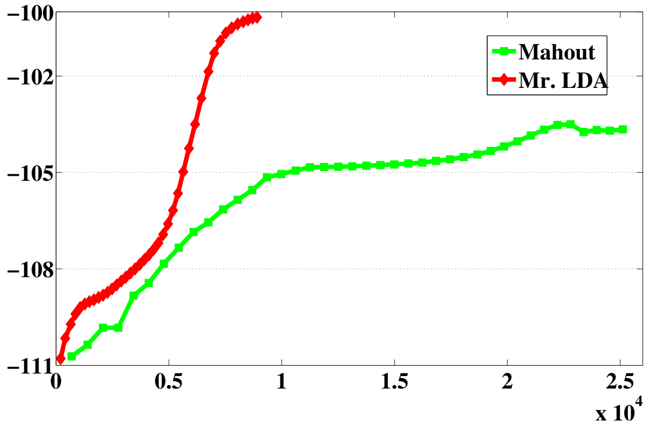


Figure 4: Training Time vs. Held-out Likelihood on 50 **topics. This figure shows the accumulated training time of the model against the held-out likelihood over Mr. LDA and Mahout. The time and held-out likelihood are measured over** 40 **iterations with** 50 **topics. Markers indicate the finishing point of a iteration. Mr. LDA out-performs Mahout both in speed and performance.**

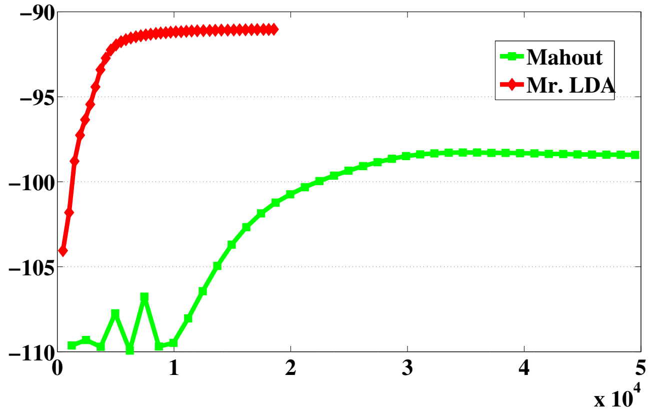


Figure 5: Training Time vs. Held-out Likelihood on 100 **topics. Similar to Figure 4, this figure shows the accumulated training time of the model against the held-out likelihood for Mr. LDA and Mahout over** 40 **iterations, but for** 100 **topics. Markers indicate the finishing point of a iteration.**

[5] D. Falush, M. Stephens, and J. K. Pritchard, "Inference of population structure using multilocus genotype data: linked loci and correlated allele frequencies." *Genetics*, vol. 164, no. 4, pp. 1567–1587, 2003.

[6] J. Boyd-Graber and P. Resnik, "Holistic sentiment analysis across languages: Multilingual supervised latent Dirichlet allocation," in *Proceedings of Emperical Methods in Natural Language Processing*, 2010.

[7] T. L. Griffiths, M. Steyvers, D. M. Blei, and J. B. Tenenbaum, "Integrating topics and syntax," in *Proceedings of Advances in Neural Information Processing Systems*, 2005.

[8] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in *Symposium on Operating System Design and Implementation*, San Francisco, California, 2004, pp. 137–150.

[9] C. Dyer, A. Cordova, A. Mont, and J. Lin, "Fast, easy and cheap: Construction of statistical machine translation models with MapReduce," in *Workshop on SMT (ACL 2008)*, Columbus, Ohio, 2008.

[10] T. Brants, A. C. Popat, P. Xu, F. J. Och, and J. Dean, "Large language models in machine translation," in *Proceedings of Emperical Methods in Natural Language Processing*, 2007.

[11] S. B. Cohen and N. A. Smith, "Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction," in *Conference of the North American Chapter of the Association for Computational Linguistics*, 2009.

[12] D. Mimno, H. Wallach, J. Naradowsky, D. Smith, and A. McCallum, "Polylingual topic models," in *Proceedings of Emperical Methods in Natural Language Processing*, 2009, IR, p. 880–889.

[13] A. K. McCallum, "Mallet: A machine learning for language toolkit," 2002, http://www.cs.umass.edu/ mccallum/mallet.

[14] F. Yan, N. Xu, and Y. Qi, "Parallel inference for latent dirichlet allocation on graphics processing units," in *Proceedings of Advances in Neural Information Processing Systems*, 2009, pp. 2134–2142.

[15] A. Asuncion, P. Smyth, and M. Welling, "Asynchronous distributed learning of topic models," in *Proceedings of Advances in Neural Information Processing Systems*, 2008.

[16] R. Nallapati, W. Cohen, and J. Lafferty, "Parallelized variational EM for latent Dirichlet allocation: An experimental evaluation of speed and scalability," in *ICDMW*, 2007.

[17] Y. Wang, H. Bai, M. Stanton, W.-Y. Chen, and E. Y. Chang, "PLDA: parallel latent Dirichlet allocation for large-scale applications," in *International Conference on Algorithmic Aspects in Information and Management*, 2009.

[18] A. J. Smola and S. Narayanamurthy, "An architecture for parallel topic models," *International Conference on Very Large Databases*, vol. 3, 2010.

[19] A. S. Foundation, I. Drost, T. Dunning, J. Eastman, O. Gospodnetic, G. Ingersoll, J. Mannix, S. Owen, and K. Wettin, "Apache Mahout," 2010, http://mloss.org/software/view/144/.

[20] R. M. Neal, "Probabilistic inference using Markov chain Monte Carlo methods," University of Toronto, Tech. Rep. CRG-TR-93-1, 1993.

[21] C. Robert and G. Casella, *Monte Carlo Statistical Methods*, ser. Springer Texts in Statistics. New York, NY: Springer-Verlag, 2004.

[22] Y. W. Teh, "A hierarchical Bayesian language model based on Pitman-Yor processes," in *Proceedings of the Association for Computational Linguistics*, 2006.

[23] T. L. Griffiths and M. Steyvers, "Finding scientific topics," *Proceedings of the National Academy of Sciences*, vol. 101, no. Suppl 1, pp. 5228–5235, 2004.

[24] J. R. Finkel, T. Grenager, and C. D. Manning, "The infinite tree," in *Proceedings of the Association for Computational Linguistics*, 2007.

[25] D. Newman, A. Asuncion, P. Smyth, and M. Welling, "Distributed Inference for Latent Dirichlet Allocation," in *Proceedings of Advances in Neural Information Processing Systems*, 2008.

[26] A. Ahmed, M. Aly, J. Gonzalez, S. Narayanamurthy, and A. Smola, "Scalable inference in latent variable models," in *WSDM*, 2012, pp. 123–132.

[27] L. Yao, D. Mimno, and A. McCallum, "Efficient methods for topic model inference on streaming document collections," in *Knowledge Discovery and Data Mining*, 2009.

[28] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *Machine Learning*, vol. 37, no. 2, pp. 183–233, 1999.

[29] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference," *Foundations and Trends in Machine Learning*, vol. 1, no. 1–2, pp. 1–305, 2008.

[30] D. M. Blei and M. I. Jordan, "Variational inference for Dirichlet process mixtures," *Journal of Bayesian Analysis*, vol. 1, no. 1, pp. 121–144, 2005.

[31] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei, "Hierarchical Dirichlet processes," *Journal of the American Statistical Association*, vol. 101, no. 476, pp. 1566–1581, 2006.

[32] K. Kurihara, M. Welling, and N. Vlassis, "Accelerated variational Dirichlet process mixtures," in *Proceedings of Advances in Neural Information Processing Systems*, Cambridge, MA, 2007.

[33] J. Wolfe, A. Haghighi, and D. Klein, "Fully distributed EM for very large datasets," in *Proceedings of International Conference of Machine Learning*, 2008, pp. 1184–1191.

[34] H. Wallach, D. Mimno, and A. McCallum, "Rethinking LDA: Why priors matter," in *Proceedings of Advances in Neural Information Processing Systems*, 2009.

[35] T. White, *Hadoop: The Definitive Guide (Second Edition)*, 2nd ed., M. Loukides, Ed. O'Reilly, 2010.

[36] A. Asuncion, M. Welling, P. Smyth, and Y. W. Teh, "On smoothing and inference for topic models," in *Proceedings of Uncertainty in Artificial Intelligence*, 2009.

[37] T. P. Minka, "Estimating a dirichlet distribution," Microsoft, Tech. Rep., 2000, http://research.microsoft.com/en-us/um/people/minka/papers/dirichlet/.

[38] J. Lin and C. Dyer, *Data-Intensive Text Processing with MapReduce*, ser. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2010.

[39] C. Lin and Y. He, "Joint sentiment/topic model for sentiment analysis," in *Proceedings of the ACM International Conference on Information and Knowledge Management*, 2009.

[40] J. W. Pennebaker and M. E. Francis, *Linguistic Inquiry and Word Count*, 1st ed. Lawrence Erlbaum, August 1999.

[41] D. M. Blei and J. D. McAuliffe, "Supervised topic models," in *Proceedings of Advances in Neural Information Processing Systems*. MIT Press, 2007.

[42] J. Boyd-Graber and D. M. Blei, "Syntactic topic models," in *Proceedings of Advances in Neural Information Processing Systems*, 2008.

[43] NIST, "Trec special database 22," 1994, http://www.nist.gov/srd/nistsd22.htm.

[44] M. Koppel, J. Schler, S. Argamon, and J. Pennebaker, "Effects of age and gender on blogging," in *In AAAI 2006 Symposium on Computational Approaches to Analysing Weblogs*, 2006.

[45] D. Hu and L. K. Saul, "A probabilistic model of unsupervised learning for musical-key profiles," in *International Society for Music Information Retrieval Conference*, 2009.

[46] G. Maskeri, S. Sarkar, and K. Heafield, "Mining business topics in source code using latent dirichlet allocation," in *ISEC*, 2008.

[47] D. Talbot and M. Osborne, "Smoothed bloom filter language models: Tera-scale lms on the cheap," in *ACL*, 2007, pp. 468–476.

[48] J. Winn and C. M. Bishop, "Variational message passing," *Journal of Machine Learning Research*, vol. 6, pp. 661–694, 2005.

[49] M. Hoffman, D. M. Blei, and F. Bach, "Online learning for latent dirichlet allocation," in *NIPS*, 2010.