Sys-TM: A Fast and General Topic Modeling System

Yingxia Shao, *Member, IEEE* Xupeng Li, Yiru Chen, Lele Yu, and Bin Cui, *Senior Member, IEEE*

Abstract—Topic models, such as LDA and its variants, are popular probabilistic models for discovering the abstract "topics" that occur in a collection of documents. However, the performance of topic models may vary a lot for different workloads, and it is not a trivial task to achieve a well-optimized implementation. In this paper, we systematically study all recently proposed samplers over LDA: AliasLDA, F+LDA, LightLDA, and WarpLDA, and discover a novel system tradeoff by considering the diversity and skewness of workloads. Then, we propose a hybrid sampler which can cleverly choose an efficient sampler with the tradeoff, and apply the hybrid sampler to LDA and its variants, including STM, TOT and CTM. Finally, we build a fast and general topic modeling system Sys-TM, which provides a unified topic modeling framework by integrating the hybrid sampler. Based on our empirical studies, the hybrid sampler outperforms the state-of-the-art samplers by up to $2\times$ over various topic models, and with carefully engineered implementation, Sys-TM is able to outperform the existing systems by up to $10\times$.

Index Terms—Topic Model, Sampling Method, Algorithm Optimization, Hybrid

1 Introduction

Topic model, as a famous probabilistic model, is to discover abstract "topics" that occur in a collection of documents. The last decade has witnessed a wide array of applications of such technique like marketing analysis [1], social event analysis [2], [3], image processing [4], [5], user modeling [6], [7] and software engineering [8]. Latent Dirichlet Allocation (LDA) is the most famous topic model, and abundance of systems are designed for LDA, including LightLDA [9], YahooLDA [10]. However, none of these systems is designed to deal with diversity and skewness of workloads that we see from real-world datasets. On the other hand, a variety of topic models have been proposed as well. Taking some typical models as examples, Supervised Topic Model (STM) [11] is an LDA variant which assumes that each document is attached with a response variable so that it can take more information about documents into account; Topic over Time (TOT) [12] model assumes each topic is associated with a continuous distribution over time to capture how the popularity of topics changes over time; and Correlated Topic Model (CTM) [13] uses the correlation matrix of Gaussian distribution to model the correlation between topics. But none of the existing systems is designed to deal with LDA variants. In this paper, we ask *how to design a single system to efficiently support a variety of topic models with the consideration of the diversity and skewness of workloads*.

Challenge 1: the diversity and skewness of datasets. In many applications, users need to run topic modeling on documents of very different types such as web corpus, user behaviors logs, and social network posts. Therefore, the length of documents varies—the average length of the NIPS dataset is about 1,000 tokens while it is only 90 for the PubMED dataset. More significantly, the word frequency in most natural language corpus follows a power law distribution [14] where some words are orders of magnitude more frequent than others. This diversity and skewness of input corpus cause different systems to slow down on different datasets we are trying to support. Worse, there are no studies of this tradeoff available so far.

Challenge 2: the diversity of topic models. Some topic models are constructed with different distributions, e.g., Dirichlet distribution in LDA and Logistic Normal distribution in CTM, while some ones require different formats of datasets, e.g., STM and TOT require documents with labels and timestamps, respectively. To design a general framework that can be easily extended to inference different topic models with high performance, a systematic study over various topic models and their inference algorithms is needed. Although the algorithms for LDA have been widely studied, there are still few studies on inference algorithms for other topic models and none of the state-of-the-art algorithms for LDA can be applied to them directly.

Motivated by these two challenges, we build a fast and general topic modeling system, Sys-TM. It not only considers the diversity and skewness in datasets, but also supports various other topic models. Our approach is to systematically study the system tradeoff caused by the diversity and skewness of the workloads, unify a variety of topic models and their inference algorithms, and design general topic modeling framework accordingly. With the integration of the recent exciting advancement of fast inference algorithms for LDA model and carefully engineered implementation, Sys-TM can be up to $10\times$ faster than the existing systems and allow users to easily implement their own new high-performance topic models.

Overview of Technical Contributions

Gibbs sampling is the *de facto* algorithm to solve topic

- *Yingxia Shao is with Beijing Key Lab of Intelligent Telecommunications Software and Multimedia, School of Computer Science, Beijing University of Posts and Telecommunications. Xupeng Li, Yiru Chen, and Bin Cui are with the Key Lab of High Confidence Software Technologies (MOE), School of EECS, Peking University, China. Lele Yu is with Tencent Inc., China.*
  E-mail: shaoyx@bupt.edu.cn, {lixupeng, chen1ru, bin.cui}@pku.edu.cn, leleyu@tencent.com

modeling that has been used intensively by previous work [9], [10], [15], [16], [17], [18], [19]. We define a corpus $\mathcal{C} = \{D_1, ..., D_n\}$ as a set of documents, and each document D_i contains a set of tokens $\{t_{ij}\}$, and let $\mathcal{T}$ be the set of all tokens. Let $\mathcal{V}$ be the vocabulary, a set of distinct words that a token can take value from, and K be the number of topics, which is usually a hyperparameter to the system. The goal of topic modeling is to assign for each token t_{ij} a topic distribution—one probability value for each of the K topics. The data structures involved in topic modeling can be represented with two matrices: (1) $\mathbf{C_w} : \mathcal{V} \mapsto \mathbb{N}_+^K$ assigns each word to a topic distribution, and (2) $\mathbf{C_d} : \mathcal{C} \mapsto \mathbb{N}_+^K$ assigns each document to a topic distribution. With Gibbs sampling, the order of assigning topics to tokens varies among different models. Typically, samplers can apply doc-first order that sampling all tokens of one document before another, or word-first order, vice versa.

1. Sampler Selection. We systematically study four samplers of LDA model: AliasLDA, F+LDA, LightLDA, and WarpLDA [9], [17], [18], [19], and focus on the question that "Out of the four recently proposed Gibbs samplers, which one should we select given a corpus?" As we will see, a suboptimal selection of a sampler can be 5× slower than the optimal strategy on LDA.

Sparsity-Aware (SA) Samplers. AliasLDA, and F+LDA exploit the sparse structure of $\mathbf{C_w}$ and $\mathbf{C_d}$ to reduce the sampling complexity. Traditionally SA samplers use word-first order, and the complexity is $O(K_d)$ for AliasLDA and F+LDA, where $K_d = |C_d(d)|$ is the number of topics ever picked by at least one token in a document. When the length of documents is skewed, the performance is dominated by the longest document.

Metropolis-Hastings (MH) Samplers. LightLDA and WarpLDA use Metropolis-Hastings to achieve an $O(1)$ sampling complexity. The time complexity per token of an MH sampler is orthogonal to the document length, the word frequency or the number of topics. However, MH samplers usually require more than one sample for each token.

System Tradeoff. We discover a tradeoff between SA samplers and MH samplers. The distributions of document lengths and word frequencies are the keys. In word-first order, when the datasets contain many long documents, SA samplers can be 2× slower than MH samplers and the performance gap can become more significant when we increase the number of topics; on the other hand, MH samplers need more iterations to converge to a comparable solution—for dataset with many short documents, MH samplers can be 1.5-5× slower than SA samplers. In doc-first order, a similar tradeoff can be observed under the word frequency distribution. Besides, a similar phenomenon holds for the number of topics K.

Hybrid Sampler. The study of the tradeoff raises a question that "Can we design a hybrid sampler that outperforms both SA and MH samplers?" The answer is yes. We propose a simple, but novel hybrid sampler based on F+LDA and WarpLDA for topic models. The empirical results in Section 6 show that the hybrid sampler can consistently outperform all others over all of our datasets by up to 2×.

2. Topic Model Revised. Besides LDA, we focus on three other popular topic models, STM, TOT and CTM. We study how to apply state-of-the-art samplers of LDA to them.

We summarize two rules for applying SA and MH samplers to topic models. (1) In order to apply SA sampler (F+LDA), we need to guarantee the computation independence between different topics, i.e., given two topics k_1 and k_2, the computation of topic-related probabilities should share no common variables. (2) For MH sampler (WarpLDA), it has to draw samples from two proposal distributions q^{doc} and q^{word}. We only need to guarantee that q^{doc} is independent of words and q^{word} is independent of documents. Therefore, we can simply set q^{doc} and q^{word} based on corresponding full conditional distribution.

Then we revise STM by following the idea of accepting a sample with certain probability from Metropolis-Hastings algorithm for approximation, so that the revised STM satisfies rule (1) and can apply SA sampler. The evaluation results show that the approximation does help STM benefit from the SA sampler. And TOT and CTM are revised as well based on the proposed rules.

3. Sys-TM Implementation and Empirical Evaluation. We develop Sys-TM based on the study of system tradeoffs and various topic models. Along with the previous two technical contributions, we also describe engineering considerations involved in building such a system. With the systematical study of various topic models, we find that most of topic modeling procedures are similar to the EM evaluation framework. With this observation, we build Sys-TM on top of a general topic modeling framework, which contains sampling step and estimating step. Meanwhile Sys-TM is equipped with a user-friendly programming interface to allow user developing new high-performance topic models flexibly. Furthermore, we conduct comprehensive experiments with a set of real and large datasets—on these datasets, Sys-TM can be up to 10× faster than the existing systems.

This paper extends a preliminary work [20] in the following aspects. First, we systematically study a range of topic models and the sampling-based inference algorithms over these models. Second, we discover system tradeoffs in all topic models we discussed and generalize the hybrid sampler to all these models. Third, we implement a fast and general topic modeling system Sys-TM based on our hybrid samplers. Fourth, we conduct extensive experiments to verify the performance improvement of the hybrid samplers over LDA variants.

2 Preliminary

We start by describing topic modeling with LDA and how to solve it with Gibbs sampling. We then present background on the basis of Metropolis-Hastings, a general case of Gibbs sampling that many state-of-the-art algorithms used to further optimize their samplers.

2.1 Latent Dirichlet Allocation

LDA is a generative probabilistic model that runs over a collection of documents (or other discrete sets). A *corpus* $\mathcal{C}$ is a collection of D documents $\{d_1, \cdots, d_D\}$ and each *document* d_i is a sequence of L_i tokens denoted by $d_i = (t_1, t_2, \cdots, t_{L_i})$, where each *token* t_n is a *word*. Each *word* w is an item from a vocabulary set $\mathcal{V}$. Let K be the

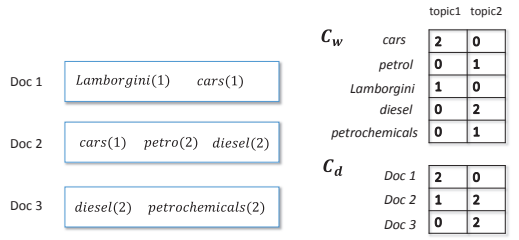


Fig. 1. An example for Corpus. The numbers in the parentheses are the current topic of each token. $\mathbf{C_w}$ and $\mathbf{C_d}$ contain counts of tokens that belong to a specific topic.

number of topics, LDA models each document as random mixtures over latent topics. Formally, each document d is represented as K-dim topic distribution θ_d which follows a Dirichlet prior α, while each topic k is a V-dim word distribution ϕ_k which follows a Dirichlet prior β. LDA generates a document d in the following steps:

1) Draw topic distribution $\theta_d \propto Dir(\alpha)$.
2) For each token,
   a) Draw topic assignment $z_{dn} \propto \mathbf{Mult}(\theta_d)$.
   b) Draw word $t_{dn} \propto \mathbf{Mult}(\phi_{z_{dn}})$.

We further denote $\mathbf{Z} = \{\mathbf{z}_d\}_{d=1}^{D}$ as the topic assignments for all tokens, where $\mathbf{z}_d = \{z_{dn}\}_{n=1}^{L_d}$, and $\mathbf{\Phi} = [\phi_1 \cdots \phi_V]$ be the topic word matrix with $V \times K$ dimensions. We use $\mathbf{\Theta} = [\theta_1 \cdots \theta_D]$ to denote the $D \times K$ document topic matrix. The inference process of LDA is to obtain the posterior distribution of latent variables $(\mathbf{\Theta}, \mathbf{\Phi}, \mathbf{Z})$ given observations $\mathcal{C}$ and hyperparameters α and β. Collapsed Gibbs Sampling (CGS) integrates out $(\mathbf{\Theta}, \mathbf{\Phi})$ through conjugacy and iteratively samples z_{dn} for tokens from the following full conditional distribution:

$$p(z_{dn} = k | t_{dn} = w, \mathbf{Z}_{\neg dn}, \mathcal{C}_{\neg dn}) \propto \frac{C_{wk}^{\neg dn} + \beta}{C_k^{\neg dn} + V\beta} (C_{dk}^{\neg dn} + \alpha) \quad (1)$$

where C_{dk} is the number of tokens that are assigned to topic k in document d; C_{wk} is the times that word w is assigned to topic k; $C_k = \sum_w C_{wk} = \sum_d C_{dk}$. The superscript or subscript $\neg dn$ represents that z_{dn} or t_{dn} is excluded from the count value or collections. Moreover, we use $\mathbf{C_w}$ to represent the $V \times K$ matrix formed by all C_{wk} and $\mathbf{C_d}$ to stand for the $D \times K$ matrix formed by all C_{dk}. $\mathbf{C_w}[w,]$ and $\mathbf{C_d}[d,]$ represent the particular rows indexed by w and d. Fig. 1 gives a simple example for a corpus with three documents.

Core Operation. During the inference phase of LDA, CGS iteratively assigns topics for tokens in $\mathcal{C}$. For one token, it calculates out all probabilities for K topics according to Eq. 1 and then randomly picks a new one. We call this process the *core operation*. This induces an $O(K)$ computation complexity per token and is very inefficient for applications with massive tokens and large value of K. For more details, readers can refer [21]. After burn-in, CGS is able to generate samples that follow the posterior distribution $p(\mathbf{Z}|\mathcal{C}, \alpha, \beta)$. We can use these samples to estimate the distribution of $\mathbf{Z}$, $\mathbf{\Theta}$ and $\mathbf{\Phi}$, which allow us to understand the semantic information of documents and words.

The process of Gibbs sampling is to run many, often tens or hundreds of, epochs over the dataset. Each epoch executes the core operation to sample its topic assignment for each token and estimates the distribution of $\mathbf{Z}$, $\mathbf{\Theta}$ and $\mathbf{\Phi}$ using the generated samples.

Algorithm 1: Metropolis-Hastings algorithm

Input : $p(x)$, $q(x)$, number of steps M
Initialize $x^{(0)} \sim q(x)$
for $i \leftarrow 1$ **to** M **do**
  Propose $x^{cand} \sim q(x^{(i)}|x^{(i-1)})$
  Acceptance rate $\pi = min\{1, \frac{p(x^{cand})q(x^{(i-1)}|x^{cand})}{p(x^{(i-1)})q(x^{cand}|x^{(i-1)})}\}$
  if *Uniform(0,1)* $\leq \pi$ **then** $x^{(i)} = x^{cand}$
  else $x^{(i)} = x^{(i-1)}$

2.2 Metropolis-Hastings

Directly sampling from probability distribution of Eq.1 is expensive. Here we describe an efficient method to reduce the sampling complexity, which is called *Metropolis-Hastings* (MH) algorithm.

Let $p(x)$ be the target distribution we want to draw samples from. MH method constructs a Markov chain with an easy-to-sample proposal distribution $q(x)$. Starting with an arbitrary state $x^{(0)}$, MH repeatedly generates samples from the proposal distribution $x^{(i)} \sim q(x^{(i)}|x^{(i-1)})$ at each step i, and updates the current state with the new sample with an *acceptance rate* $\pi = min\left\{1, \frac{p(x^{(i)})q(x^{(i-1)}|x^{(i)})}{p(x^{(i-1)})q(x^{(i)}|x^{(i-1)})}\right\}$.

Algorithm 1 presents the detail of Metropolis-Hastings. We call the hyperparameter M an *MH steps*. Under certain technical condition, $q(x^{(i)})$ converges to $p(x)$ as $i \to \infty$, regardless of $x^{(0)}$ [22]. Gibbs sampling is a special case of Metropolis-Hastings.

3 Sampler Selection Over LDA

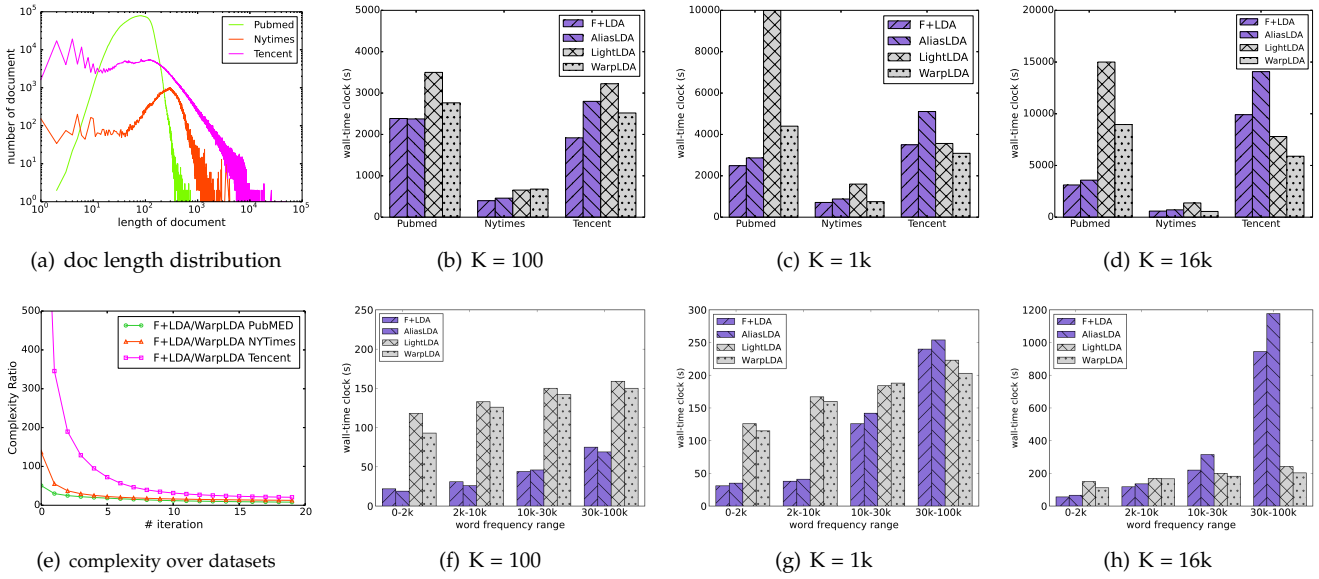
We study the system tradeoff of four recently published samplers over LDA. We start by presenting a taxonomy of four existing samplers for LDA, all of which were published recently. We study the system tradeoff of different samplers over LDA. We start by presenting a taxonomy of four existing samplers for LDA, all of which were published recently.

3.1 Anatomy of Existing Samplers

Existing samplers can be classified into two categories according to the optimizations they use to reduce the sampling complexity: (1) **SA samplers**, including AliasLDA and F+LDA, exploit the sparse structure of $\mathbf{C_d}$ or $\mathbf{C_w}$; and (2) **MH samplers**, including LightLDA and WarpLDA, use Metropolis-Hastings method to scale to a large number of topics. We study their tradeoff with the following experiment setup.

Settings. We implement all samplers under the same code base. For fair comparison, we optimize all these techniques and they are faster than all their original open-source implementations[1]. Because samplers in the original open-source codes use different implementations, e.g., different hash table implementations. In our code base, we use the same implementation in all samplers, and optimize them with more efficient data structures, like using array instead of the hash table in WarpLDA. All experiment results that

1. https://github.com/Microsoft/LightLDA,http://bigdata.ices.utexas.edu/software/nomad/,https://github.com/thu-ml/warplda



(a) doc length distribution (b) K = 100 (c) K = 1k (d) K = 16k

(e) complexity over datasets (f) K = 100 (g) K = 1k (h) K = 16k

Fig. 2. Effects of document length, word frequency, and K on the performance of samplers.

we report are on a single machine and all samplers are parallelized with 16 threads.

Metrics. We measure the performance by the wall-clock time a sampler requires to converge to a given log likelihood value. We vary the number of topics and use datasets that have different characteristics to measure each sampler's performance.

Datasets. We use three different datasets to illustrate the tradeoff. One key characteristic turns out to be the length of a document, and the document length distribution is showed in Figure 2(a). We see that these datasets have different length distribution— the PubMED dataset consists of many short documents whose length is less than 200, while the NYTimes dataset contains both short documents and long documents whose length can vary from 100 to 2000. For the Tencent dataset (the one from our industry partner), most of the tokens are from long documents with length larger than 1000—about 24% of documents have a length over 1000, which contains more than 76% of all tokens. Furthermore, the word frequency of documents usually follows power-law. The detailed statistics of the three datasets are listed in Table 1.

3.1.1 Sparse-Aware (SA) Samplers

AliasLDA and F+LDA decompose the probability for each token (Eq. 1) into two parts: $C_{dk}\frac{C_{wk}+\beta}{C_k+V\beta}$ and $\alpha\frac{C_{wk}+\beta}{C_k+V\beta}$. When C_{dk} ($\mathbf{C_d}$) is sparse, the sampler can skip those with zero C_{dk}, thus lowering the complexity. The difference between these two algorithms is the different data structures they use to perform sampling—AliasLDA uses the Alias table [23], while F+LDA uses the F+ tree.

Besides above decomposition method, Eq. 1 can also be divided in a different way. The above two parts become $C_{wk}\frac{C_{dk}+\alpha}{C_k+V\beta}$ and $\beta\frac{C_{dk}+\alpha}{C_k+V\beta}$. Under this new decomposition, SA samplers can skip zero elements when C_{wk} ($\mathbf{C_w}$) is sparse.

The above two decomposition methods for SA samplers imply two different data accessing order. We named them **word-first order** and **doc-first order** respectively. With the doc-first order, the sampler draws topics for all tokens in one document before turning to another, while with the word-first order, the sampler draws topics for all tokens associated with one word at one time. As briefly described, the different orders utilize the sparsity of different data structures, like $\mathbf{C_d}$ or $\mathbf{C_w}$.

Results. Fig. 2(b), 2(c), and 2(d) show the results for different number of topics K. In these three experiments, SA samplers are executed in word-first order. We can see that F+LDA is consistently faster than AliasLDA over all datasets and various values of K. The relative performances of AliasLDA and F+LDA are different across different datasets. Specifically, the gap becomes larger on the Tencent dataset, while it is much smaller on the other two datasets. This is because AliasLDA requires both traversal access and random access for $\mathbf{C_d}$ matrix, while F+LDA only requires traversal access. Since $\mathbf{C_d}$ is stored in sparse format—which aims to reduce memory cost—the latency of random access increases with more items in one document due to the increased conflicts in the hashtable.

Fig. 2(f), 2(g), and 2(h) show the results for different number of topics K and different word frequencies of datasets. Here we run SA samplers through the doc-first order. Moreover, we split the NYTimes dataset according to the frequency of words. Specifically, we divide the original datasets into four subsets, where each one contains words belong to different frequency ranges, including $\{[0, 2k), [2k, 10k), [10k, 30k), [30k, 100k)\}$. Then we run SA samplers over all these subsets. We see that F+LDA is faster than AliasLDA over all datasets when K is large, and it has comparable efficiency with AliasLDA when K is small (e.g., K=100). The gap of performance is still caused by the access speed of data structures employed by them.

3.1.2 Metropolis-Hastings (MH) Samplers

LightLDA and WarpLDA draw samples from two proposal distributions — $q^{doc} \propto C_{dk} + \alpha$ and $q^{word} \propto \frac{C_{wk}+\beta}{C_k+\beta}$ —

alternatively, and accept their proposals based on the acceptance condition of Metropolis-Hastings. These two samplers are different in their access order for the in-memory data structures—WarpLDA separates the access of tokens into two passes, where each pass only accesses $\mathbf{C_w}$ or $\mathbf{C_d}$, respectively, in order to reduce random access.

Results. Fig. 2(b), 2(c), 2(d) show the results of MH Samplers for different kinds of datasets. We see that WarpLDA is consistently faster than LightLDA. This is because WarpLDA effectively eliminates the amount of random accesses. The performance gap is influenced by the size of the vocabulary $|\mathcal{V}|$ — the more words the corpus has, the more random accesses will be incurred by $\mathbf{C_w}$. For the PubMED dataset, which has the largest vocabulary, the performance gap between WarpLDA and LightLDA is largest; on the other hand, since the Tencent dataset only contains 88,916 distinct words in its vocabulary, this performance gap is relatively small.

Fig. 2(f), 2(g), and 2(h) show the results of MH Samplers for different number of topics K and different word frequencies of datasets. We see that WarpLDA is still faster than or at least comparable with LightLDA. This is also because WarpLDA eliminates the amount of random access. In addition, the datasets generated by splitting NYTimes have relatively small vocabulary sizes, the performance gap between WarpLDA and LightLDA is not significant.

3.2 Tradeoff: SA vs. MH Samplers

We now describe the system tradeoff between SA and MH samplers. Because in our experiments F+LDA and WarpLDA always dominate others, we focus on the tradeoff between them.

Summary of the Tradeoff

Length of Document L_d. The length of documents turns out to be an important axis in the tradeoff when SA samplers run following the word-first order. On datasets with many short documents, like PubMED, F+LDA is faster than WarpLDA—by up to 2.8× when $K = 16k$; On datasets with many long documents, like Tencent, WarpLDA is faster than F+LDA— by up to 1.7× when $K = 16k$.

Frequency of Word L_w. The frequency of words presents another important factor in the tradeoff when SA samplers run following the doc-first order. On datasets containing many low-frequency words, like subset with word frequency ranging from 0 to $2k$, F+LDA is faster than WarpLDA—by up to 3.5× when $K = 1k$; On datasets with many high-frequency words, WarpLDA is faster than F+LDA—by up to 4.7× when $K = 16K$.

Number of topics K. The number of topics also plays a major role in the tradeoff. When the number of topics increases—e.g., from $1k$ to $16k$ on Tencent— WarpLDA becomes faster compared with F+LDA; Similarly, when the number of topics increases on PubMED, F+LDA becomes faster compared with WarpLDA. For doc-first experiments, F+LDA is faster than WarpLDA on all subsets when $K = 100$. When the number of topics increase—e.g. $1k$ and $16k$, WarpLDA shows better performance for subsets with high-frequency words.

Analysis. The above tradeoff can be explained with the following analytic model. To generate one sample in WarpLDA, the Metropolis-Hastings sampler needs to decide whether to accept a sample or not. Let π be the acceptance rate, WarpLDA requires, in expectation, $\frac{1}{\pi}$ samples for one sample to be accepted. On the other hand, F+LDA does not have this overhead. However, to generate each sample, the computational complexity of F+LDA is $O(K_d)$ in word-first order, where K_d, the number of non-zero elements of C_d, is bound by the number of topics K and the length of a document L_d, or $O(K_w)$ in doc-first order, where K_w, the number of non-zero elements of C_w, is bounded by the number of topics K and the frequency of word L_w. On the other hand, WarpLDA incurs $O(1)$ complexity to propose each sample.

We can now see the tradeoff—when K, L_d, and L_w are small, WarpLDA is slower because of its $\frac{1}{\pi}$ overhead; otherwise, F+LDA is slower due to its overhead in generating each sample.

Figure 2(e) further provides a quantitative illustration of the tradeoff. We define a *complexity ratio* $\lambda = K_d\pi$ as the ratio between the complexity of F+LDA and WarpLDA. As expected, on datasets where F+LDA is faster, λ is smaller. This is consistent with the empirical result illustrated in Figure 2(e).

3.3 Hybrid Sampler

The above tradeoff raises a natural question— *Can we build a hybrid sampler that marries both SA and MH sampler?* Intuitively, this is trivial—just run "short" documents with F+LDA and "long" documents with WarpLDA in word-first order, or run "rare" words with F+LDA and "common" words with WarpLDA in doc-first order. In practice, however, there are two technical questions: (1) how to decide which document is *long enough* or word is *common enough* to "qualify" for WarpLDA; and (2) how to *balance* an MH sampler and an SA sampler, which have different convergence speeds. For the first question, we develop a very simple rule-of-thumb based on the study of our tradeoff. The second question is more challenging and ties to an open question in the mixing theory of two Markov chains. Inspired by classic statistical theory on a simpler underlying model, we develop a simple heuristics that works well across all of our datasets.

3.3.1 Sampler Selection Strategy

The first step is to design a rule-of-thumb to choose between two samplers. We focus on the combination of F+LDA and WarpLDA, as these two samplers dominate consistently faster than AliasLDA and LightLDA in our tradeoff study.

Sampling Complexity. For F+LDA in word-first order to generate one sample, it needs to traverse the non-zero items in C_{dk}, whose size is bounded by K and L_d. And in doc-first order, it traverses the non-zero items in C_{wk}, whose size is bounded by K and L_w. Thus, the sampling complexity of F+LDA is

$$C_{f+} = O(min(L_d, K)) \;\; or \;\; O(min(L_w, K)).$$

On the other hand, since Metropolis-Hastings method requires mix time for each sample, the complexity of WarpLDA is

$$C_{warp} = O(n),$$

where n is the steps to achieve mix.

The technical challenge is how to estimate n, whose value not only depends on the input datasets, but also changes across iterations. In theory, estimating the *mixing time* of the underlying MCMC chain for general factor graphs is a problem that has been around for decades. With this in mind, we resort to a simple heuristics that is motivated by the empirical observation from our study.

Heuristics 1 (word-first order). Given a document d with length L_d and topic number K, if $K \leq S$ or $L_d \leq S$ choosing F+LDA; otherwise, choosing WarpLDA. S is a threshold value that depends on the implementation and properties of the dataset.

Heuristics 2 (doc-first order). Given a word w with frequency L_w and topic number K, if $K \leq S$ or $L_w \leq S$ choosing F+LDA; otherwise, choosing WarpLDA. S is a threshold value that depends on the implementation and properties of the dataset.

3.3.2 Balancing Two Chains

One surprising observation is that the above heuristics itself is not enough for a hybrid sampler that outperforms both F+LDA and WarpLDA. The fundamental reason is that two Markov chains underlying F+LDA and WarpLDA *do not converge with the same speed*. Specifically, all samples from F+LDA will be accepted, however this is not the case for WarpLDA. Intuitively, this means that a naive hybrid sampler would generate more samples for tokens belonging to F+LDA per epoch than WarpLDA.

Theoretical Understanding The above observation raises a fundamental question: *What is the relationship between the mixing time of a Gibbs sampler (F+LDA) and a Metropolis-Hastings (WarpLDA) sampler for the same underlying distribution?* If we *magically* know the ratio between their mixing time, we could just use this number to balance these two chains.

Unfortunately, a general treatment of this question has existed for decades and is still under intensive study by the theoretical computer science and mathematics community. However, there are theoretical results that can be used to *inspire* our practical heuristics.

We know that the mixing time of these two chains is not too different, at least for the Ising model:

Lemma 3.1. [22](Levin, Peres, & Wilmer 2008, Example 13.18) For a graph with vertex set V, let π be the Ising probability measure. Let γ be the spectral gap of the Gibbs chain and $\tilde{\gamma}$ be the spectral gap of the Metropolis chain using the base chain, we have $\gamma \leq \tilde{\gamma} \leq 2\gamma$, where γ is related to the mixing time by the following lemma

Lemma 3.2. [22](Levin, Peres, & Wilmer 2008, Theorem 12.3) Let P be the transition matrix of a reversible, irreducible Markov chain with state space Ω, and π be the underlying probability measure. Let $\pi_{min} = \min_{x\in\Omega} \pi(x)$ and γ the absolute spectral gap (equals to the spectral gap when the chain is "lazy"), we have the mixing time

$$t_{min}(\epsilon) \leq \log(\frac{1}{\epsilon\pi_{min}})\frac{1}{\gamma} \tag{2}$$

The above two lemmas inspired the design in two ways. First, we know, at least intuitively, whatever scheme we use to balance these two chains, that scheme should not be "too extreme" (these two chains are off by a constant factor anyway). Second, the mixing time is likely to be linear (or inverse linear) to the constant that we will use to balance these two chains. Inspired by these, our simple heuristics is as follows:

Heuristics 3. For each epoch, the MH steps for the WarpLDA is set to $\lceil \frac{1}{\pi} \rceil$, where π is the acceptance rate for the last epoch.

The intuition behind this heuristics is simple—just use the empirical acceptance rate $1/\pi$ as the proxy for the ratio of spectral gap $\frac{\tilde{\gamma}}{\gamma}$ (In extreme cases where all MH samples are accepted ($\pi = 1$), MH becomes Gibbs, and therefore $\frac{\tilde{\gamma}}{\gamma} = \frac{1}{\pi}$ in this extreme case.)

4 Samplers For LDA Variants

After designing the hybrid sampler over LDA, a natural question is *Can we use the hybrid sampler to improve the performance of LDA variants?* The answer is yes, and we should apply SA and MH sampler to LDA variants. According to the above experiments, F+LDA and WarpLDA always dominate other samplers, therefore, we focus on applying F+LDA and WarpLDA to LDA variants. For simplicity, in the context of LDA variants, SA sampler stands for F+LDA and MH sampler means WarpLDA.

Considering that the tradeoff in hybrid sampler has been fully discussed in the previous section. Here we only elaborate the rules of applying SA and MH samplers, and the corresponding revisions of LDA variants, including STM, TOT, and CTM.

4.1 The Rules of Applying SA and MH Samplers

All the full conditional distributions (Eq. 1, 3, 4, 5) of LDA and its variants have factor $C_{dk}+\alpha$ or $C_{wk}+\beta$, for simplicity, the distributions can be denoted as $(C_{dk}+\alpha)F(w,k)$ or $(C_{wk}+\beta)F(d,k)$. Since $F(w,k)$ and $F(d,k)$ are similar, the following discussion takes $F(w,k)$ as example.

Rule for SA sampler. When using the SA sampler to improve sampling performance, the key technique is applying F+Tree to maintain $F(w,k)$ for each topic k, and decompose the probability into $C_{dk}F(w,k) + \alpha F(w,k)$ to skip those with zero C_{dk}. To correctly run SA sampler, the following condition must hold: *Given any two topics k_1, k_2, $F(w,k_1)$ and $F(w,k_2)$ should share no common variables to be calculated.* Taking LDA as an example, $F(w,k) = \frac{C_{wk}+\beta}{C_k+V\beta}$, where C_{wk} and C_k are variables, V and β are constants. It is easy to figure out that $F(w,k_1)$ and $F(w,k_2)$ share no common variables. This is because C_{wk_1} and C_{wk_2} are different, so are C_{k_1} and C_{k_2}.

Rule for MH sampler. In Section 3.1.2 we have introduced that the MH sampler has to draw samples from two proposal distributions — q^{doc} and q^{word}, and accept their proposals based on the acceptance condition of Metropolis-Hastings. To apply the MH sampler, we need to guarantee: 1) q^{doc} is independent of words and 2) q^{word} is independent of documents. On basis of this rule, we can simply set q^{doc} to be $(C_{dk}+\alpha)$ and q^{word} to be $F(w,k)$.

4.2 Samplers for Supervised Topic Model

Compared with LDA, Supervised Topic Model (STM) [11] adds a response variable to each document. It models documents and response variables in a jointly way in order to explore topic distributions more precisely, with predicting the labels of new documents at the same time.

The difference of generating one document between STM and LDA is that STM needs to draw one response variable y from a normal distribution after drawing all words of this document. Besides that, other processes remain the same as LDA. The generative process of STM can be summarized as follows:

1) Draw topic distribution $\theta_d \propto Dir(\alpha)$.
2) For each token:
   a) Draw topic assignment $z_{dn} \propto \mathbf{Mult}(\theta_d)$.
   b) Draw word $t_{dn} \propto \mathbf{Mult}(\phi_{z_{dn}})$.
3) Draw response variable $y \propto \mathcal{N}(\eta^T \overline{z_d}, \sigma^2)$, where $\overline{z_d} = \frac{1}{L_d}\sum_{n=1}^{L_d} z_{dn}$ is the empirical topic frequency and z_{dn} is of one-hot representation.

To inference an STM model, we need to obtain the posterior distribution of latent variables $(\mathbf{\Theta}, \mathbf{\Phi}, \mathbf{Z})$ as well as the parameters (η, σ^2) of the normal distribution for response variables. The EM algorithm is employed here to accomplish the inference process. In each iteration of this algorithm, there are two steps:

1) Applying CGS to sample z_{dn} for each token from the following full conditional distribution:

$$p(z_{dn}=k|t_{dn}=w,\eta,\sigma^2,\mathbf{Z}_{\neg dn},\mathcal{C}_{\neg dn}) \propto (C_{dk}^{\neg dn}+\alpha)\ \frac{C_{wk}^{\neg dn}+\beta}{C_k^{\neg dn}+V\beta}\ exp(-\frac{1}{2\sigma^2}[\frac{\eta_k}{L_d}(2\eta^T\overline{z_d^{\neg n}}-2y_d-\frac{\eta_k}{L_d})]) \tag{3}$$

2) Calculating the optimal value of η to maximize the log-likelihood $log\ p(\mathbf{Z},\mathbf{W},\mathbf{Y}|\alpha,\beta,\eta,\sigma^2)$ by estimating the expectation of $\mathbf{Z}$ using the sampling result z_{dn}.

$$log\ p(\mathbf{Z},\mathbf{W},\mathbf{Y}|\alpha,\beta,\eta,\sigma^2) \propto \sum_{d=1}^{D} log\ \mathcal{N}(y_d|\eta^T\overline{z_d},\sigma^2).$$

We denote the optimal value of η as $\eta^* = (\mathbf{Z}^T\mathbf{Z})^{-1}\mathbf{Z}^T\mathbf{Y}$.

SA sampler. The distribution of STM model in Eq. 3 almost conforms to $(C_{wk}+\beta)F(d,k)$ if we set $F(d,k)$ to be $\frac{C_{dk}+\alpha}{C_k+V\beta}\ exp(-\frac{1}{2\sigma^2}[\frac{\eta_k}{L_d}(2\eta^T\overline{z_d^{\neg n}}-2y_d-\frac{\eta_k}{L_d})])$, so we can optimize Gibbs sampling with the sparsity of word distribution in the doc-first order.

However, the above $F(d,k)$ does not satisfy the rule of applying SA sampler. This is because $\overline{z_d^{\neg n}}$ makes $F(d,k)$ shares common variables over k. Specifically, $\overline{z_{dk}^{\neg n}}$ equals $C_{dk}^{\neg n}/L_d$, and the values $C_{dk}^{\neg n}$ of all k are required by each $F(d,k)$. Therefore, F+Tree in F+LDA cannot be used to maintain the values $F(d,k)$ directly.

To eliminate the shared variables, we borrow the idea of accepting a sample with certain probability from MH sampler, and propose an approximation of $F(d,k)$.

Assume $\overline{z_d^0}$ is the value when document d to be sampled, we revise the Eq. 3 by using $\overline{z_d^0}$ to replace $\overline{z_d^{\neg n}}$, then the approximated $F(d,k)$ is

$$\tilde{F}(d,k) = \frac{C_{dk}+\alpha}{C_k+V\beta}\ exp(-\frac{1}{2\sigma^2}[\frac{\eta_k}{L_d}(2\eta^T\overline{z_d^0}-2y_d-\frac{\eta_k}{L_d})])$$

To remedy the deviation between $F(d,k)$ and $\tilde{F}(d,k)$, we add an accept step where acceptance rate is:

$$min(1, exp\{\frac{1}{\sigma^2}\frac{\eta_k-\eta_{k_0}}{L_d}\eta(\overline{z_d^{\neg n}}-\overline{z_d^0})\})$$

With above revisions, we can consider $\overline{z_d^0}$ as constants in the sampling step and apply F+LDA sampler to STM model.

MH Sampler. With the new version of full conditional distribution $(C_{wk}+\beta)\tilde{F}(d,k)$, we can set $q^{word} \propto C_{wk}+\beta$ and $q^{doc} \propto \tilde{F}(d,k)$, and the acceptance rates are $\pi^{word} = min(1, \frac{F(d,k)}{F(d,k_0)})$ and

$$\pi^{doc} = min(1, \frac{C_{wk}+\beta}{C_{wk_0}+\beta}exp\{\frac{1}{\sigma^2}\frac{\eta_k-\eta_{k_0}}{L_d}\eta(\overline{z_d^{\neg n}}-\overline{z_d^0})\}).$$

Therefore, the sampler can draw topics in average $O(1)$ complexity.

4.3 Samplers for Topics over Time Model

Topics over Time model (TOT) [12] plays an important role on discovering topic evolution over time. This model assumes a continuous distribution over time associated with each topic. Meanwhile, topics are responsible for generating both observed timestamps and words. Under TOT model, each document arises from the following generative process:

1) Draw topic distribution $\theta_d \propto Dir(\alpha)$.
2) For each token:
   a) Draw topic assignment $z_n \propto \mathbf{Mult}(\theta)$.
   b) Draw word $t_{dn} \propto \mathbf{Mult}(\phi_{z_{dn}})$.
   c) Draw timestamp $s_{dn} \propto Beta(\psi_{z_n,1}, \psi_{z_n,2})$.

where $Beta(\psi_{z_n,1}, \psi_{z_n,2})$ represents Beta distribution with hyperparameter Ψ=($\psi_{z_n,1}$, $\psi_{z_n,2}$).

Different from STM which draws one response variable for each document, TOT draws a timestamp for each token from a Beta distribution with hyperparameter Ψ based on the associated topic assignment.

The inference algorithm for TOT is also an EM based algorithm which conducts sampling and estimating parameters iteratively.

1) In the sampling step, we apply CGS which integrates out $\mathbf{\Theta}$ and $\mathbf{\Phi}$ through conjugacy to sample z_{dn} for each token from the full conditional distributions.

$$p(z_{dn}=k|s_{dn},\mathbf{Z}_{\neg dn},\mathcal{C}) \propto (C_{dk}^{\neg dn}+\alpha)\frac{C_{wk}^{\neg dn}+\beta}{C_k^{\neg dn}+V\beta}\ \frac{s_{dn}^{-1+\psi_{z_{dn},1}}(1-s_{dn})^{-1+\psi_{z_{dn},2}}}{B(\psi_{z_{dn},1},\psi_{z_{dn},2})} \tag{4}$$

where $B(\alpha,\beta) = \Gamma(\alpha)\Gamma(\beta)/\Gamma(\alpha+\beta)$.
2) In the estimating step, we calculate the optimal value of Ψ to maximize the log likelihood $log\ p(\mathbf{Z},\mathbf{W},\mathbf{S}|\Psi,\alpha,\beta)$.

$$log\ p(\mathbf{Z},\mathbf{W},\mathbf{S}|\Psi,\alpha,\beta) \propto \sum_{d=1}^{D}\sum_{i=1}^{L_d} log\ Beta(t_{di}|\psi_{z_{di},1},\psi_{z_{di},2})$$

$$\psi_{z,1} = \overline{s_z}(\frac{\overline{s_z}(1-\overline{s_z})}{S_z^2}-1), \qquad \psi_{z,2} = \frac{1-\overline{s_z}}{\overline{s_z}}\psi_{z,1}$$

where $\overline{s_z}$ and S_z^2 indicate the sample mean and the biased sample variance of the timestamps belonging to topic z, respectively.

SA sampler. To adapt SA sampler to TOT model, we observe that the distribution of TOT in Eq. 4 has the form $(C_{wk}+\beta)F(d,k)$ where

$$F(d,k) = \frac{C_{dk}+\alpha}{C_k+V\beta}\frac{{s_{di}}^{\psi_{k,1}-1}(1-s_{di})^{\psi_{k,2}-1}}{B(\psi_{k,1},\psi_{k,2})}.$$

Note that the timestamps s_{di} of one document are the same values and can be considered as a constant s_d of that document. Therefore, we can optimize Gibbs sampling with the sparsity of word distribution in the doc-first order.

MH Sampler. Adapting MH sampler to TOT model, we set proposal distributions $q^{word} \propto C_{wk}+\beta$ and $q^{doc} \propto F(d,k)$, and the acceptance rates are $\pi^{word} = min(1, \frac{F(d,k)}{F(d,k_0)})$ and $\pi^{doc} = min(1, \frac{C_{wk}+\beta}{C_{wk_0}+\beta})$.

4.4 Samplers for Correlated Topic Model

Correlated Topic Model (CTM) [13] captures the correlation between topics by adopting logistic normal distribution for topic distribution. Given one document d, the generative process is listed as follows:

1) Draw topic distribution $\theta_d \propto \mathcal{N}(\mu,\Sigma)$.
2) For each token:
   a) Draw topic assignment $z_{dn} \propto \mathbf{Mult}(f(\theta_d))$ where $f(\theta_i) = exp(\theta_i)/\sum_j exp(\theta_j)$.
   b) Draw word $t_{dn} \propto \mathbf{Mult}(\phi_{z_{dn}})$.

In CTM, the topic distribution θ_d is drawn from a normal distribution with hyperparameters μ and Σ. Because normal distribution $\mathcal{N}(\mu,\Sigma)$ and multinomial distribution $\mathbf{Mult}(f(\theta_d))$ are not conjugate, it is impossible to integrate out $\mathbf{\Theta}$ for the topic distribution. Therefore, we can only apply the original Gibbs sampling rather than applying Collapsed Gibbs Sampling algorithm for sampling $\mathbf{Z}$. To sample the continuous random variable $\mathbf{\Theta}$, we apply Stochastic Gradient Langevin Dynamics (SGLD) algorithm [24]. Since the documents are independent with each other in SGLD, we implement the SGLD in data parallel, i.e., in each iteration, SGLD computes $\Delta\theta_i$ and updates θ_i for a batch of documents in parallel.

Finally, with Gibbs sampling, we iteratively sample $\mathbf{Z}$ and $\mathbf{\Theta}$ from the following distributions:

1) Sample $\mathbf{Z}$ in the similar way with LDA's:

$$p(z_{dn}=k|t_{dn},\mathbf{Z}_{\neg dn},\mathcal{C}_{\neg dn},\mathbf{\Theta}) \propto \frac{C_{wk}^{\neg dn}+\beta}{C_k^{\neg dn}+V\beta}\, e^{\theta_{dk}} \quad (5)$$

2) Sample $\mathbf{\Theta}$ by SGLD algorithm:

$$p(\mathbf{\Theta}|\mathbf{Z},\mathcal{C}) \propto \prod_{d=1}^{D}\{\mathcal{N}(\theta_d|\mu,\Sigma)\prod_{i=1}^{L_d}\frac{e^{\theta_{d,z_{di}}}}{\sum_k e^{\theta_{d,k}}}\}$$

SA sampler. The distribution of CTM in Eq. 5 also conforms $(C_{wk}+\beta)F(d,k)$ where

$$F(d,k) = \frac{e^{\theta_{dk}}}{C_k+V\beta},$$

so we can adapt SA sampler to CTM model with the doc-first order.

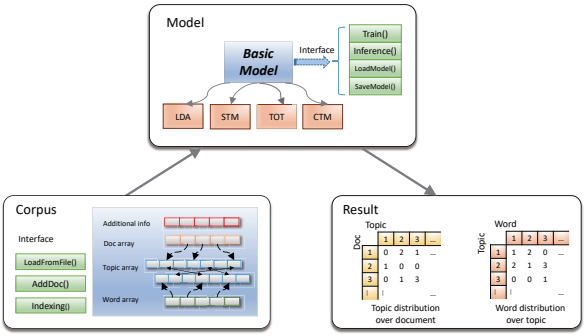


Fig. 3. The high-level abstraction of Sys-TM.

MH Sampler. Similar with previous two adaptions, we set proposal distributions $q^{word} \propto C_{wk}+\beta$ and $q^{doc} \propto F(d,k)$, and the acceptance rates are $\pi^{word} = min(1, \frac{F(d,k)}{F(d,k_0)})$ and $\pi^{doc} = min(1, \frac{C_{wk}+\beta}{C_{wk_0}+\beta})$.

4.5 Sampling Complexity of LDA variants

Due to the "rule for SA sampler", F+LDA runs in the doc-first order over LDA variants. Thus, the sampling complexity of F+LDA is

$$C_{f+} = O(min(L_w, K)).$$

On the other hand, unlike LDA (Section 4.3 in [19]), the topic assignment of LDA variants cannot use Random positioning, and needs Alias sampling to achieve $\mathcal{O}(1)$ time to propose a sample, which entails $\mathcal{O}(K)$ initialization complexity for each document in every iteration. Therefore, the complexity of WarpLDA is

$$C_{warp} = O(n + \frac{K}{L_d}),$$

where n is the steps to achieve mix.

5 Sys-TM: A Topic Modeling System

In this section, we present the details of our open-source topic modeling system, Sys-TM[2], which is carefully implemented with previous technique contributions. The system is designed based on a general topic modeling framework, which integrates the hybrid sampler to guarantee the high-performance of topic models across different workloads. On top of the framework, we have implemented all the topic models discussed previously — LDA, STM, TOT and CTM. In addition, Sys-TM allows users to implement their own high-performance topic models with friendly interfaces.

5.1 Architecture of Sys-TM

Figure 3 shows the high-level abstraction of Sys-TM. It mainly consists of three components: Corpus, Model, and Result. The details of each component are described as below.

Corpus A Corpus is a set of documents which contains a number of tokens. Inside a Corpus, each document and each token have an unique integer identifier, and everything is stored with those integer IDs. Specially, tokens are stored

2. https://github.com/DMALab/Sys-TM

in either doc-first order or word-first order, which is determined by the user so that the inference algorithm can traverse the corpus in a proper order to achieve the best cache efficiency. Refer to our previous work [20] for the detailed data structures for long or short docs which guarantees the high performance no matter what the corpus or samplers are. To support LDA variants, documents can be further labeled with additional information such as timestamps or category IDs.

Model A Model is a prototype of topic models. The prototype can also be called basic model. The basic model records a set of parameters and topic-word counters that can be used to inference new corpus, and it provides several interfaces: *Train(Corpus)*, *Inference(Corpus)*, *SaveModel(path)*, and *LoadModel(path)*.

A customized topic model can be instantiated by implementing the interfaces with its own model parameters and inference algorithms. To facilitate customization, in Sys-TM we implement the prototype as an efficient hybrid sampler template. The sampler template integrates all the technique contributions, and users can easily write a new sampling algorithm by implementing interfaces of defining related distributions in the template, i.e., filling up the formulas $F(d, k)$ or q^{word} in a new model. In Sys-TM, we have implemented samplers for LDAModel, STMModel, TOTModel and CTMModel on the basis of this template. To train a topic model, users can create a new model instance, specify the values of model parameters, and invoke $Train()$ interface over a corpus. After training, you can get the topic-word distributions, which is the trained model. Typically, the topic-word and doc-topic distributions are stored as counter matrices C_{wk} and C_{dk} as described in Section 2. Considering sparsity, we treat C_{wk} (in doc-first order) or C_{dk} (in word-first order) as CSC sparse matrix. Then, users can apply the trained model on a new Corpus to inference doc-topic distributions or save the trained model for future usage. In addition, users can do incremental training in Sys-TM. This means that users can update the existing trained model with new corpus for online training [25]. What's more, the Train and Inference methods are implemented with high performance hybrid samplers as we have mentioned before.

Result A Result keeps doc-topic distributions after the training or inference process. Users can easily access the results or save them into a file.

5.2 Topic Modeling Framework

Considering that most of topic models do sampling from a posterior distribution and then estimate parameters, we design the topic modeling framework for Sys-TM following the computation architecture in expectation maximization (EM) algorithm. More specifically, our topic modeling framework has two steps: (1) Sampling step: it samples a value from a posterior distribution; (2) Estimating step: it estimates parameters with sampled values. To clarify, the estimating step can be omitted if there are no parameters to be estimated, like LDA and CTM.

In a sampling step, an algorithm usually samples topics for all tokens in a given document set, and also sample other model-specific variables from continuous distributions, like $\boldsymbol{\Theta}$ in STM. To sample topics, we apply our hybrid samplers by considering the sparsity of a model. We split the whole document set into two parts based on document lengths or word frequencies and then sample tokens in the two parts separately with F+LDA sampler or WarpLDA sampler. To sample continuous variables, we apply SGLD algorithm [26]. In an estimating step, the algorithm computes the optimal parameter values to maximize the log likelihood of current sampling results. The algorithm repeats sampling step and estimating step until the log likelihood converges.

To implement a customized topic model on top of our framework, users only need to extend basic model by defining model specific parameters, like the traverse order of F+LDA sampler, and implementing their own model-specific functions. These functions include sampling variables from continuous distributions, estimating parameters, and user-defined parts in hybrid sampler.

6 Evaluation

In this section, we report experimental results of three parts: (1) we show results validating our technical contribution of system tradeoffs; (2) we evaluate our hybrid samplers and compare them with state-of-the-art samplers over LDA and its variants; (3) we compare our Sys-TM with lightLDA, Gensim library and other open-sourced implementations of LDA variants.

6.1 Experimental Setup

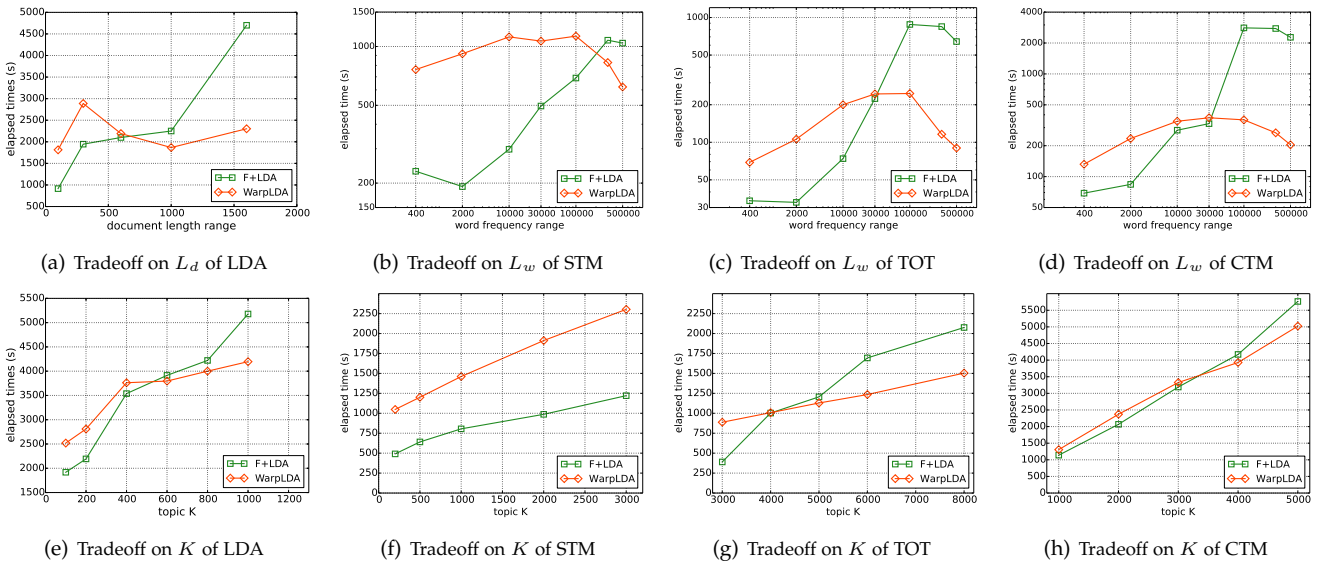
Experiment Setting. We conduct all single-machine experiments on a server which has an Intel Xeon E5530 @ 2.4G CPU with 16 cores, 74GB DDR3 memory and 10×2TB SATA hard disks. Following prior arts, we set the hyperparameter of Dirichlet distribution $\alpha = \frac{50.0}{K}$ and $\beta = 0.01$ for all the following experiments. All samplers are re-implemented in Sys-TM.

TABLE 1
Dataset Statistics.

dataset	#docs	#words	#tokens	average doc length	average word frequency
NYTimes	300K	102660	99M	331	964
PubMED	8.2M	141044	737M	90	5225
Tencent	2.5M	88916	1.26B	504	14170
BaiKe	2.8M	98234	418M	148	4225
PubMED*	300K	97994	19M	88	193
BaiKe*	1M	88834	148M	148	1666

Datasets. Table 1 summarizes the datasets we used to evaluate the speed of samplers. They are NYTimes, PubMED, Tencent, and BaiKe. Compared with the other three datasets, BaiKe is a dataset with many long documents as well as many short documents. NYTimes and PubMED are public datasets[3], while the other datasets are from our industry partner. Due to the high consumption of STM and CTM models when K is large, we further randomly sample 300K documents from PubMED and 1M documents from BaiKe, and generate two small datasets, PubMED* and BaiKe*.

3. https://archive.ics.uci.edu/ml/machine-learning-databases/bag-of-words/



(a) Tradeoff on L_d of LDA (b) Tradeoff on L_w of STM (c) Tradeoff on L_w of TOT (d) Tradeoff on L_w of CTM

(e) Tradeoff on K of LDA (f) Tradeoff on K of STM (g) Tradeoff on K of TOT (h) Tradeoff on K of CTM

Fig. 4. Tradeoff points of L_d, L_w and K of LDA, STM, TOT, CTM models.

Metrics. We measure the quality of a topic model by the log joint likelihood, and count the wall-clock time for a method to reach a given log likelihood value in order to compare their performance. In each experiment, we stop training and record the log likelihood value when the difference of values between two consecutive iterations is no more than 0.1%.

6.2 System Tradeoffs

We now validate that all L_d, L_w, and K which forms a tradeoff between SA and MH samplers in all four topic models. We show the tradeoffs of L_d and K in LDA when running SA with word-first order, and tradeoffs of L_w and K in STM, TOT, and CTM when running SA with doc-first order.

Tradeoff of L_d. To determine the tradeoff point of L_d in LDA, we construct five datasets with different document lengths by separating the Tencent. The first dataset consists of documents with $L_d \in [0, 0.2k)$, while the second one contains documents with $L_d \in [0.2k, 0.4k)$. The lengths of documents for the other three datasets are $[0.4k, 0.6k)$, $[0.6k, 1.2k)$, and $[1.2k, 2k)$. We run F+LDA and WarpLDA with $K = 8000$ on these five datasets and their wall-clock time to reach the same convergent log likelihood are shown in Figure 4(a). From this figure, we can see that on dataset with $L_d \in [0, 0.2k)$, F+LDA is 2× faster than WarpLDA, while F+LDA is 2× slower than WarpLDA on the dataset with $L_d \in [1.2k, 2k)$. For the dataset containing documents with $L_d \in [0.4k, 0.6k)$, F+LDA and WarpLDA present nearly the same performance. Therefore, in our hybrid sampler, we use F+LDA for documents with length less than 600 and choose WarpLDA otherwise.

Tradeoff of L_w. To determine the tradeoff point of L_w in LDA variants, we construct seven datasets with different word frequencies from NYTimes. The frequencies of L_w for them are $[0, 0.4k)$, $[0.4k, 2k)$, $[2k, 10k)$, $[10k, 30k)$, $[30k, 100k)$, $[100k, 300k)$ and $[300k, 500k)$. We run F+LDA and WarpLDA with $K = 8000$ in TOT model, $K = 3000$ in STM and CTM model on these datasets and their wall-clock times to reach the same convergent log likelihood are shown in Figures 4(b), 4(c) and 4(d).

From the figure of TOT, we can see that on dataset with $L_w \in [0, 0.4k)$, F+LDA is 2× faster than WarpLDA, while F+LDA is 4× slower than WarpLDA on the dataset with $L_w \in [30k, 100k)$. For the dataset containing words with $L_w \in [10k, 30k)$, F+LDA and WarpLDA present nearly the same performance. CTM model has similar tradeoff with TOT, while STM model has a much higher cross point. For STM model, F+LDA is faster than WarpLDA in the first five datasets ($L_w < 100k$), and WarpLDA eventually outperforms F+LDA with large L_w. This is because when L_w is small, the cost of STM is dominated by the computation of η^*, which is not influenced by the variable L_w.

Finally, in TOT and CTM samplers, we use F+LDA for words with frequency less than $30k$. Otherwise, we choose WarpLDA. In STM sampler, the tradeoff point is $200k$.

Tradeoff of K. To determine the tradeoff point of K in LDA model, we evaluate both F+LDA and WarpLDA on the Tencent dataset with K range from 100 to 1,000. The wall-clock time to reach the same log likelihood value is presented in Figure 4(e). We can see that F+LDA can outperform WarpLDA when K is small, while WarpLDA is faster when K becomes larger. The cross point happens when $K \simeq 600$ in our experiments. Therefore, in our implementation of the hybrid sampler in LDA model, we use F+LDA when K is less than 600; otherwise, we use WarpLDA.

To determine the tradeoff point of K in STM, TOT, and CTM models, we evaluate both F+LDA and WarpLDA on NYTimes dataset with K range from 100 to 8000. The wall-clock time to reach the same log likelihood value is presented in 4(f) 4(g) 4(h). For TOT and CTM models, we observe results that F+LDA outperforms WarpLDA when K is small and WarpLDA is better when K is large. For STM model, F+LDA always outperforms WarpLDA. As mentioned in the previous discussion of L_w, the cost of STM is dominated by computing η^* with small L_w, so

TABLE 2
Time (seconds) for samplers of LDA variants to reach the same log likelihood. Note that * in column **K&L** indicates that PubMED* or BaiKe* is used.

dataset	K& L	MH	SA	hybrid	K& L	MH	SA	hybrid	K& L	MH	SA	hybrid
	STM				TOT				CTM			
Tencent	1k&1.1e9	4242	2568	2518	3k&7.2e9	5121	9892	3315	1k&-6.6e9	4404	10612	4169
	2k&1.9e9	5518	3211	2854	5k&1.3e10	7436	23079	3979	2k&9.7e8	5632	13613	5139
	3k&3.0e9	6776	4851	3457	8k&2.2e10	12492	50773	6167	3k&1.6e9	7642	18429	6569
PubMED	1k&1.2e9*	435	133	124	3k&1.3e11	11550	5748	5278	1k&1.6e8*	1103	923	829
	2k&2.9e9*	666	145	192	5k&2.36e11	18150	7457	6286	2k&5.9e8*	1451	1619	1584
	3k&4.8e9*	1144	432	322	8k&4.1e11	28195	4272	7742	3k&1.0e9*	2765	2025	2347
BaiKe	1k&7.2e9	6882	2022	3796	3k&3.6e10	5442	12332	3483	1k&-1.7e9*	4021	5008	3024
	2k&2.0e10	7123	7533	3830	5k&6.7e10	8335	15970	4140	2k&-1.3e*	3053	10180	2296
	3k&3.4e10	9991	13734	5371	8k&1.1e11	11792	12974	4153	3k&-1.1e9*	6681	16590	5188

TABLE 3
Time (seconds) for samplers of LDA to reach the same log likelihood.

dataset	K& L	SA		MH		hybrid
		F+	Alias	Light	Warp	
PubMED	1k&-6.6e9	2489	2865	11024	4400	2322
	8k&-6.7e9	2851	3355	13045	7389	2212
	16k&-6.8e9	3115	3580	15426	8957	2713
Tencent	1k&-9e9	3500	5108	3564	3087	2142
	8k&-9e9	7746	11839	7500	4747	4341
	16k&-9e9	9910	14072	7800	5894	5773
BaiKe	8k&-3.6e9	1893	2505	3308	2045	1158
	16k&-3.6e9	2763	3065	6930	3792	1779

that the costs of one iteration of F+LDA and WarpLDA are similar, but F+LDA needs fewer iterations than WarpLDA to converge. Based on our experiment results, we set the tradeoff points at 4000 in TOT model and 3000 in CTM model, and STM has no tradeoff point of K.

6.3 The Performance of Hybrid Samplers

We now validate that advantages of our hybrid sampler. Tables 3 and 2 give a summary of the performance for all samplers by presenting their wall-clock time to a given value of log likelihood. Due to the constrained space, we do not list the results on NYTimes in the tables, and actually, the results are similar to other datasets.

Hybrid sampler on LDA. The SA sampler in LDA model runs in word-first order. We see that our hybrid sampler outperforms the fastest existing samplers on all datasets. On BaiKe, which contains both short documents and long documents, our sampler can outperform both F+LDA and WarpLDA by 1.7-2.1×. This is because our sampler is able to choose the most appropriate sampler to use. For PubMED dataset, our hybrid sampler is 2-3× faster than WarpLDA, while having comparable performance to F+LDA. For Tencent dataset, our hybrid sampler is 1.7× faster than F+LDA and is comparable with WarpLDA. For the detailed analysis of other samplers, please refer to Section 3.1.

Hybrid sampler on LDA variants. We test our hybrid sampler on LDA variants, and the SA sampler runs in doc-first order. The results in Table 2 show that our hybrid sampler outperforms the fastest existing samplers on most experiments. On Tencent and Baike dataset, which contain documents with both low and high word frequency, our hybrid sampler can outperform both SA and MH sampler by up to 2.7×. For PubMED dataset, our hybrid sampler is up to 3.6× faster than MH sampler, while having comparable performance to (or slightly slower than) SA sampler. On the basis of complexity analysis in Section 4.5, the efficiency of WarpLDA on LDA variants is related to the number of topics and the length of documents. Specifically, when the topic number is much larger than the document length, the cost of initialization of WarpLDA cannot be omitted. Therefore, on PubMED, whose average length of documents is short (~ 90), the hybrid sampler may not be the best one with a large number of topics (e.g., K=2000). The results also imply that there are complex influences of L_d, L_w and K for the efficiency of hybrid sampler over LDA variants, and the simple heuristics may fail to select the appropriate sampler. However, due to the difficulty of mixing time estimation, it is non-trivial to quantify the complex influences, and we leave it as an open question.

TABLE 4
Time (seconds) comparison among Sys-TM, **open-sourced** F+LDA, WarpLDA, LightLDA and Gensim.

K	F+ LDA	Light LDA	Warp LDA	Gensim	Sys-TM
0.1k	495	525	271	12244	87
1k	547	1196	410	21848	186
3k	536	1335	431	>10hrs	209

6.4 End-to-End Performance of Sys-TM

In this section, we compare Sys-TM with popular open-sourced Topic modeling libraries. All the experiments are conducted on NYTimes with topic number 100, 1000 and 3000.

Table 4 shows the results of comparing Sys-TM with four state-of-the-art LDA libraries, and they are LightLDA, WarpLDA, F+LDA, and Gensim [4] on a single machine. For Sys-TM, LightLDA, WarpLDA and F+LDA, we report the wall-time of all samplers to achieve the same value of log likelihood. The log likelihood values are -9.7e8, -9.8e8 and -1.0e9 for K=0.1k, K=1K and K=3K, respectively. We can see that WarpLDA performs the best among all three open-source LDA algorithms and Sys-TM with hybrid sampler can still have over 2× speed up. For Gensim, we find Sys-TM is remarkably faster than Gensim by approximately 100∼150×. The main reason is that the complexity of hybrid sampler in Sys-TM is much lower than that of Gensim. Gensim uses variational inference, and it costs $\Omega(K)$ for each token, where K is the number of topics, in one iteration. While Sys-TM costs at most $O(K_d)$, where K_d is the average number of none-zero values in C_d which is much smaller than K. Further, when WarpLDA is selected by our

4. https://github.com/Microsoft/LightLDA, https://github.com/thu-ml/warplda, http://bigdata.ices.utexas.edu/software/nomad/, https://radimrehurek.com/gensim/

TABLE 5
Time comparison among Sys-TM and **open-sourced** STM, CTM and TOT.

K	STM Blei-lab	Sys-TM	CTM Blei-lab	Sys-TM	TOT	Sys-TM
0.1k	$\gg$10hrs	902	$\gg$10hrs	878	$\gg$10hrs	853
1k	$\gg$10hrs	1984	$\gg$10hrs	2545	$\gg$10hrs	1137
3k	$\gg$10hrs	2991	$\gg$10hrs	5958	$\gg$10hrs	1439

hybrid sampler, the cost is reduce to only $O(1)$ per token. Besides, our carefully engineered implementation of Sys-TM also boosts the improvement gains.

Table 5 shows the costs of running Sys-TM and open-sourced LDA variants, including STM, CTM and TOT[5]. We find that Sys-TM is extremely faster ($\gg 10\times$) than current open sourced LDA variants. With topic number 100, 1000 and 3000, Sys-TM finishes computation around 1 hour, while all the original versions can not finish in 10 hours. This is not only because Sys-TM applies the hybrid sampler which integrates state-of-the-art LDA samplers, but also Sys-TM benefits from the carefully implemented data structures.

7 Related Work

Topic modeling is one of the most powerful machine learning technology that has been widely used in text mining, network analysis, and many other domains. Topic models are usually probabilistic models that give an unsupervised approach to find hidden structures or semantics in gigantic information. Early topic models include LSI [27], pLSI [28], and Latent Dirichlet allocation (LDA) which is the most popular topic model currently in use. Afterwards, many topic models have been proposed to enhance various aspects of LDA. For example, D. Blei et al. proposed Correlated Topic Model (CTM) [13] to discover correlations between topics. They also proposed Supervised Topic Model (STM) [11], Labeled LDA [29], and Dynamic Topic Model (DTM) [30] to handle various kinds of information associated with documents. Wang et al. proposed Topic over Time (TOT) [12] to discover the popularity variations of topics over time. Chang et al. proposed Relational Topic Model (RTM) [31] to model relations or links between documents. Recently on the basis of CTM, Correlated Gaussian Topic Model (CGTM) [32] is proposed by leveraging external resources to model the topic correlations, and NVCTM [33] uses Centralized Transformation Flow to capture the correlations among topics by reshaping topic distributions. In this paper, we focus on providing a unified and efficient library for implementing various topics models, like LDA, STM, CTM and TOT.

Gibbs sampling is a popular method to inference probabilistic models, which is also the most efficient one to inference topic models. There are a range of Gibbs samplers that have been proposed to reduce the sampling complexity of LDA. SparseLDA [15] is the first to take advantage of the sparse structure of $\mathbf{C_d}$ and $\mathbf{C_w}$ to achieve $O(K_d + K_w)$ sampling complexity. However, for a large dataset, the value of K_w for popular words equals to K—AliasLDA [17] and F+LDA [18] further reduces the complexity to $O(K_d)$ by factorizing the posterior distribution. LightLDA [9] proposes an $O(1)$ method by alternately sampling from two simple proposals through MH method, while WarpLDA [19] reorders the sampling operations from these two proposals to reduce random memory access. Although the complexity for one sampling operation is $O(1)$ for them, we find that they need more sampling operations to generate one sample, since the MH method requires mix time. FastLDA [34] uses the skew property of Eq. 1 to calculate only a fraction of the K topic probabilities per sampling operation. In this work, we carefully studied the tradeoff among different samplers, and proposed a hybrid sampler to select a proper sampler based on the characteristics of datasets and the models.

To scale up the training of LDA for large datasets, parallelization is also a good choice. All existing parallelization methods divide documents into multiple partitions and each worker conducts sampling operations. AD-LDA [35] proposes an approximate sampling method where each worker uses the stale version of the $\mathbf{C_w}$ matrix and requires synchronization among all workers after each iteration. PLDA [36] is an MPI and MapReduce implementation of AD-LDA. PLDA+ [37] proposes to sample over word order at each worker and then utilize the pipeline method to overlap the computation and communication. YahooLDA [10] proposes a distributed key-value cache to share the $\mathbf{C_w}$ matrix among different workers. However, it samples tokens through the document order and requires maintenance of one copy of $\mathbf{C_w}$ at each worker. NomadLDA [18] utilizes the nomad token to perform asynchronous sampling while guaranteeing there will be no update conflicts. With regard to the dynamic latent topics, Clustered Latent Dirichlet Allocation (CLDA) [38], [39] significantly accelerates DTM to discover dynamic topics in a large collection of documents via parallel computing.

8 Conclusion

In this paper, we systematically studied four topic models and their sampling-based inference algorithms. Then we studied the state-of-the-art samplers for LDA and discovered a tradeoff between the Sparse-Aware sampler and Metropolis-Hastings samplers. Based on this tradeoff, we developed a hybrid sampler that employs different samplers for documents of different lengths and different word-frequency. We further adapted the hybrid sampler of LDA to inference other topic models. Finally, we built a fast and general topic modeling system, named Sys-TM, which integrates the novel hybrid sampler and is carefully implemented.

Acknowledgments

This work is supported by the National Key Research and Development Program of China (No. 2018YFB1004403), NSFC (No. 61832001, 61702015, 61572039), Beijing Academy of Artificial Intelligence (BAAI), and PKUTencent joint research Lab. Bin Cui is the corresponding author.

5. https://github.com/blei-lab/class-slda, https://github.com/blei-lab/ctm-c, https://github.com/ahmaurya/topics_over_time

REFERENCES

[1] S. Tirunillai and G. J. Tellis, "Mining marketing meaning from online chatter: Strategic brand analysis of big data using latent dirichlet allocation," *Journal of Marketing Research*, vol. 51, no. 4, pp. 463–479, 2014.

[2] S. Qian, T. Zhang, C. Xu, and M. S. Hossain, "Social event classification via boosted multimodal supervised latent dirichlet allocation," *ACM TOMM*, vol. 11, no. 2, p. 27, 2015.

[3] N. Bennacer Seghouani, F. Bugiotti, M. Hewasinghage, S. Isaj, and G. Quercini, "A frequent named entities-based approach for interpreting reputation in twitter," *Data Science and Engineering*, vol. 3, no. 2, pp. 86–100, Jun 2018.

[4] J. Wang, J. Zhou, H. Xu, T. Mei, X.-S. Hua, and S. Li, "Image tag refinement by regularized latent dirichlet allocation," *Computer Vision and Image Understanding*, vol. 124, pp. 61–70, 2014.

[5] K. S. Arun and V. K. Govindan, "A hybrid deep learning architecture for latent topic-based image retrieval," *Data Science and Engineering*, vol. 3, no. 2, pp. 166–195, Jun 2018.

[6] L. Wang, L. Zhang, and J. Jiang, "Iea: an answerer recommendation approach on stack overflow," *Science China Information Sciences*, vol. 62, no. 11, Sep 2019.

[7] Y. Xiao, X. Li, Y. Liu, H. Liu, and Q. Li, "Correlations multiplexing for link prediction in multidimensional network spaces," *Science China Information Sciences*, vol. 61, no. 11, Jun 2018.

[8] Y. Fu, M. Yan, X. Zhang, L. Xu, D. Yang, and J. D. Kymer, "Automated classification of software change messages by semi-supervised latent dirichlet allocation," *Information and Software Technology*, vol. 57, pp. 369–377, 2015.

[9] J. Yuan, F. Gao, Q. Ho, W. Dai, J. Wei, X. Zheng, E. P. Xing, T.-Y. Liu, and W.-Y. Ma, "Lightlda: Big topic models on modest computer clusters," in *WWW*, 2015, pp. 1351–1361.

[10] A. Smola and S. Narayanamurthy, "An architecture for parallel topic models," *PVLDB*, vol. 3, no. 1-2, pp. 703–710, 2010.

[11] J. D. Mcauliffe and D. M. Blei, "Supervised topic models," in *NIPS*, 2008, pp. 121–128.

[12] X. Wang and A. McCallum, "Topics over time: a non-markov continuous-time model of topical trends," in *SIGKDD*, 2006, pp. 424–433.

[13] D. M. Blei and J. D. Lafferty, "Correlated topic models," in *NIPS*, 2005, pp. 147–154.

[14] D. M. W. Powers, "Applications and explanations of zipf's law," in *NeMLaP3/CoNLL*, 1998, pp. 151–160.

[15] L. Yao, D. Mimno, and A. McCallum, "Efficient methods for topic model inference on streaming document collections," in *SIGKDD*, 2009, pp. 937–946.

[16] J. Zhu, A. Ahmed, and E. P. Xing, "Medlda: maximum margin supervised topic models," *Journal of Machine Learning Research*, vol. 13, no. Aug, pp. 2237–2278, 2012.

[17] A. Q. Li, A. Ahmed, S. Ravi, and A. J. Smola, "Reducing the sampling complexity of topic models," in *SIGKDD*, 2014, pp. 891–900.

[18] H.-F. Yu, C.-J. Hsieh, H. Yun, S. Vishwanathan, and I. S. Dhillon, "A scalable asynchronous distributed algorithm for topic modeling," in *WWW*, 2015, pp. 1340–1350.

[19] J. Chen, K. Li, J. Zhu, and W. Chen, "Warplda: A cache efficient o(1) algorithm for latent dirichlet allocation," *PVLDB*, vol. 9, no. 10, pp. 744–755, Jun. 2016.

[20] L. Yu, C. Zhang, Y. Shao, and B. Cui, "Lda*: a robust and large-scale topic modeling system," *PVLDB*, vol. 10, no. 11, pp. 1406–1417, 2017.

[21] T. L. Griffiths and M. Steyvers, "Finding scientific topics," *PNAS*, vol. 101, no. suppl 1, pp. 5228–5235, 2004.

[22] D. A. Levin, Y. Peres, and E. L. Wilmer, *Markov chains and mixing times*. American Mathematical Soc.

[23] A. J. Walker, "An efficient method for generating discrete random variables with general distributions," *TOMS*, vol. 3, no. 3, pp. 253–256, 1977.

[24] M. Welling and Y. W. Teh, "Bayesian learning via stochastic gradient langevin dynamics," in *ICML*, 2011, pp. 681–688.

[25] A. Banerjee and S. Basu, "Topic models over text streams: A study of batch and online unsupervised learning," in *SDM*, 2007, pp. 431–436.

[26] S. Patterson and Y. W. Teh, "Stochastic gradient riemannian langevin dynamics on the probability simplex," in *NIPS*, 2013, pp. 3102–3110.

[27] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *J Am Soc Inf Sci*, vol. 41, no. 6, p. 391, 1990.

[28] T. Hofmann, "Probabilistic latent semantic indexing," 1999.

[29] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning, "Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora," in *EMNLP*, 2009, pp. 248–256.

[30] D. M. Blei and J. D. Lafferty, "Dynamic topic models," in *ICML*, 2006, pp. 113–120.

[31] J. Chang and D. Blei, "Relational topic models for document networks," in *Artificial Intelligence and Statistics*, 2009, pp. 81–88.

[32] G. Xun, Y. Li, W. X. Zhao, J. Gao, and A. Zhang, "A correlated topic model using word embeddings," in *IJCAI*, 2017, pp. 4207–4213.

[33] L. Liu, H. Huang, Y. Gao, Y. Zhang, and X. Wei, "Neural variational correlated topic modeling," in *WWW*, 2019, pp. 1142–1152.

[34] I. Porteous, D. Newman, A. Ihler, A. Asuncion, P. Smyth, and M. Welling, "Fast collapsed gibbs sampling for latent dirichlet allocation," in *SIGKDD*, 2008, pp. 569–577.

[35] D. Newman, P. Smyth, M. Welling, and A. U. Asuncion, "Distributed inference for latent dirichlet allocation," in *NIPS*, 2007, pp. 1081–1088.

[36] Y. Wang, H. Bai, M. Stanton, W.-Y. Chen, and E. Y. Chang, "Plda: Parallel latent dirichlet allocation for large-scale applications," in *AAIM*, 2009, pp. 301–314.

[37] Z. Liu, Y. Zhang, E. Y. Chang, and M. Sun, "Plda+: Parallel latent dirichlet allocation with data placement and pipeline processing," *TIST*, vol. 2, no. 3, p. 26, 2011.

[38] C. Gropp, A. Herzog, I. Safro, P. W. Wilson, and A. W. Apon, "Scalable dynamic topic modeling with clustered latent dirichlet allocation (clda)," *ArXiv*, vol. abs/1610.07703, 2016.

[39] J. Sybrandt, M. Shtutman, and I. Safro, "Moliere: Automatic biomedical hypothesis generation system," in *KDD*, 2017, pp. 1633–1642.

Yingxia Shao is a Research Associate Professor in the School of Computer Science, Beijing University of Posts and Telecommunications. His research interests include large-scale graph analysis, parallel computing framework, and knowledge graph analysis.

Xupeng Li was a PhD student in the School of EECS, Peking University during this work. His research interests include formal verification, data management system, parallel computing framework, and machine learning.

Yiru Chen was a senior undergraduate student in the School of EECS, Peking University during this work. Her research interests include database system, interactive data analysis, machine learning and human-computer interaction.

Lele Yu is a researcher in Tencent Inc. He received his Ph.D from Peking University in 2018. His interests include distributed ML, big data processing and parallel computing systems, and Topic modeling.

Bin Cui is a professor in the School of EECS and Director of Institute of Network Computing and Information Systems, at Peking University. His research interests include database systems, and data mining. Prof. Cui has published more than 100 research papers, and is the winner of Microsoft Young Professorship award (MSRA 2008), and CCF Young Scientist award (2009).