TECHNICAL REPORT

TR-CS-NMSU-2017-08-21

Chuan Hu
Huiping Cao
Qixu Gong

Department of Computer Science
New Mexico State University

August 21, 2017

Sub-Gibbs Sampling: a New Strategy for Inferring LDA (theoretical proof and experiment details)

Chuan Hu Huiping Cao Qixu Gong
chu@cs.nmsu.edu, hcao@cs.nmsu.edu, qgong@cs.nmsu.edu
Dept. of Computer Science, New Mexico State University, U.S.A.

Abstract

In this technical report we present the theoretical proof and experiment details in our work, *Sub-Gibbs Sampling: a New Strategy for Inferring LDA*. Latent Dirichlet Allocation (LDA) has been widely used in text mining to discover topics from documents. One major approach to learn LDA is Gibbs sampling. The basic Collapsed Gibbs Sampling (CGS) algorithm computes conditional probabilities and draws topics for each token, thus requires $O(NZ)$ computations to learn an LDA model with Z topics from a corpus containing N tokens. Existing approaches that improve the complexity of CGS focus on reducing the factor Z.

In this work, we propose a novel and general *Sub-Gibbs Sampling* (SGS) strategy to improve the Gibbs-Sampling computation by reducing the sample space. This new strategy targets at reducing the factor N by sampling only a subset of the whole corpus. The design of the SGS strategy is based on two properties that we observe: (i) topic distributions of tokens are skewed and (ii) a subset of documents can approximately represent the semantics of the whole corpus. We prove that the SGS strategy can achieve comparable effectiveness (with bounded errors) and significantly reduce the running time compared with CGS algorithms. Extensive experiments on large real-world data sets show that the proposed SGS strategy is $2 \sim 100$ times faster than CGS and $2 \sim 5$ times faster than several state-of-the-art fast Gibbs sampling algorithms. On a large data set (PubMed), the SGS strategy can reduce the running time per iteration of CGS from 5 hours to 0.5 hours. Experimental results verify that the proposed SGS strategy can learn comparable LDA models as other Gibbs sampling algorithms.

1 Introduction and Related Works

Latent Dirichlet Allocation (LDA) is a widely used topic model ever since its introduction by Blei et al. [1]. Different methods, such as variational inference [1] and Gibbs Sampling [2], are used to learn LDA models. A basic Collapsed Gibbs Sampling (CGS) algorithm to infer the LDA model is introduced by Griffiths et al. [2]. The complexity of this CGS is $O(NZ)$ where N and Z are the total

number of observed tokens and the number of latent topics in a text corpus respectively. This $O(NZ)$ complexity makes CGS very expensive to run on large corpora.

To improve the basic CGS algorithm, many fast Gibbs sampling algorithms (e.g., [3, 4, 5, 6, 7]) have been proposed in the literature. Porteous et al. [3] propose the *FastLDA* algorithm to improve the running time (not complexity) of CGS by segmenting and rearranging the conditional probabilities. Yao et al. [4] design the *SparseLDA* algorithm and reduce the complexity to $O(N(Z_w + Z_d))$ where Z_w is the number of distinct topics that are assigned to a token w, Z_d is the number of distinct topics that are assigned to w's corresponding document d, and $Z_w + Z_d$ is generally smaller than Z in practice. *SparseLDA* utilizes an observation that word-topic and document-topic count matrices in Gibbs sampling are sparse. Li et al. [5] design an $O(NZ_d)$ approximate sampling algorithm, *AliasLDA*, by combining the Metropolis-Hasting sampling and the alias table method [8]. Yuan et al. [7] further improve *AliasLDA* by approximating more probability components and propose the *LightLDA* algorithm with an $O(N)$ complexity. Yu et al. [6] design the *F+Nomad LDA* Gibbs sampling algorithm with an $O(N \log Z)$ time complexity by utilizing the *Fenwick tree* [9] data structure. All these works focus on reducing the factor Z through efficient calculation and maintenance of conditional probabilities. Previous works [3, 4, 6] are exact Gibbs sampling, and [5, 7] are approximate sampling with Metropolis-Hasting sampling. They generally converge to the same (or very similar) results. The choice of using different algorithms to infer LDA mainly differs on efficiency.

In this paper we propose a totally new and general strategy, *Sub-Gibbs Sampling (SGS)*, to improve Gibbs sampling algorithms for LDA inference by reducing the sample space. I.e, we target at reducing the factor N. SGS is based on two properties which are not utilized in previous works.

- *Skewed topic distributions.* In LDA models, multiple occurrences of one token in a document are assigned with a few distinct topics, and the majority of the occurrences is assigned with the same topic. Note that this property is very different from the parameter-sparsity property used in previous research [5, 4, 7]. The difference is explained in more details in Section 3.1.
- *Approximate semantics.* The semantics of a corpus can be approximately represented by a small subset of the documents. In this paper "semantics" represents the set of distinct tokens (vocabulary) in a corpus. The documents in this representative subset is called *covered-documents* and the remaining documents are called *uncovered-documents*.

These two properties are presented in details in Section 3. Utilizing these two properties SGS is designed to run a base Gibbs sampling (i) on a small subset of tokens (instead of *all* the tokens); (ii) on covered-documents for more iterations than on uncovered-documents.

The contributions of this work are as follows.

- We identify two new useful properties in LDA models, the skewed topic distributions and the approximate semantics, which have not been utilized

before.

- Utilizing the two properties, we propose a novel and general Sub-Gibbs Sampling (SGS) strategy to reduce the sampling space of existing Gibbs sampling algorithms.
- We conduct extensive experiments on four real-world data sets (ranging from small to large). Comparison between the proposed SGS strategy and state-of-the-art fast Gibbs sampling algorithms demonstrates that SGS reduces the running time significantly (2∼5 times faster) and achieves similar effectiveness.

In what follows, Section 2 introduces the background of LDA and CGS. Section 3 explains the two properties and the proposed SGS strategy. Section 4 presents the experimental results and analysis. Section 5 reviews related works. Section 6 concludes this work.

2 Background and Notations

1. For each latent topic z_i, $i \in \{1, 2, \ldots, Z\}$, draw $\vec{\phi}_i \sim Dirichlet(\beta)$
2. For each document d_i, $i \in \{1, 2, \ldots, D\}$
   (a) Draw $\vec{\theta}_i \sim Dirichlet(\alpha)$
   (b) For $j \in \{1, 2, \ldots l_i\}$. l_i is the length of document d_i
      i. Draw the latent topic $z_{ij} \sim multinomial(\vec{\theta}_i)$
      ii. Draw the observed word $w_{ij} \sim multinomial(\vec{\phi}_{z_{ij}})$

Figure 1: Generative Process of LDA [1]

Latent Dirichlet Allocation (LDA) is introduced in [1] as a generative probabilistic model to learn topics in text corpora. LDA assumes that the observed tokens in each document are generated from a mixture (document-to-topic distributions θ) of several multinomial distributions (topic-to-word distributions ϕ). The detailed generative process of LDA is in Fig. 1. The Collapsed Gibbs Sampling (CGS) algorithm [2] is a widely accepted approach to estimate parameters in LDA. CGS estimates latent topics for tokens by iteratively drawing samples for each token from conditional probabilities. The CGS algorithm is shown in Fig. 2. The notations of LDA and CGS are listed in TABLE 1.

In the generative process of LDA (Fig. 1), $\vec{\phi}_i$ is a W-dimensional vector representing the probability of each token (in total W distinct tokens in the corpus) under topic z_i, and $\vec{\theta}_i$ is a Z-dimensional vector representing the probability of each topic (in total Z topics) being assigned to document d_i. The multinomial

1. For each iteration $i = 1, 2, \cdots$

(a) For each token w in each document d

i. Let z be the topic assignment to w in the previous iteration

ii. Decrement n_{dz}, n_d, n_{wz}, n_z by one

iii. Sample a new topic z for w from a multinomial distribution $p(z = k|\neg z),\ k \in \{1, 2, \ldots, Z\}$

$$p(z = k|\neg z) \propto \frac{n_{wk} + \beta}{n_k + W\beta} \cdot \frac{n_{dk} + \alpha}{n_d + Z\alpha} \quad (1)$$

iv. Increment n_{dz}, n_d, n_{wz}, n_z by one

Figure 2: Basic Collapsed Gibbs Sampling (CGS) algorithm to learn LDA [10]

distribution parameters $\vec{\theta}_i$ and $\vec{\phi}_i$ have Dirichlet priors whose hyper-parameters are α and β respectively. To generate the jth token w_{ij} in document d_i, latent topic z_{ij} is first drawn from $multinomial(\vec{\theta}_i)$, then w_{ij} is drawn from $multinomial(\vec{\phi}_{z_{ij}})$. In LDA tokens are observed variables, and other variables are latent variables.

Given the generative process in Fig. 1, Griffiths et al. [2] design a CGS algorithm (Fig. 2). CGS runs for many iterations. In each iteration, it maintains the sample counts for document d (n_d), the sample counts for topic z (n_z), the times that topic z is assigned to token w (n_{wz}), and the times of sampling topic z for document d (n_{dz}). In each iteration the topics of all the tokens are redrawn from the topics' conditional probabilities which are calculated as Eq. (1).

3 Sub-Gibbs Sampling Strategy

This section presents the strategy of ***Sub-Gibbs Sampling (SGS)***. SGS utilizes two properties in LDA models to reduce the sample space: (i) tokens have very skewed topic distribution patterns, and (ii) the semantics of a corpus can be approximated using a subset of documents. We denote these two properties as ***Skewed Topic Distributions*** and ***Approximate Semantics***. The idea of SGS is using a subset of tokens in a document and a subset of documents in a corpus to learn LDA models. The proposed SGS strategy is designed in a general way so that it can be applied to any Gibbs sampling algorithms for LDA. We explain these two properties and the corresponding algorithms in Sections 3.1 and 3.2.

3.1 SGS Utilizing Skewed Topic Distributions

We conduct a pre-study to examine the topic distributions of tokens and find some interesting patterns. In this pre-study, we run the CGS algorithm on the *NYTimes* data set (the statistics is shown in TABLE 3) with $Z = 100$. After

Symbol	Meaning
α, β	Hyper-parameters of Dirichlet distribution
$\vec{\theta}_i$	Multinomial topic distribution of document d_i
$\vec{\phi}_i$	Multinomial word distribution of topic z_i
D	# of documents
W	# of distinct tokens (vocabulary size)
N	Total # of tokens in the corpus
Z	# of latent topics
d_i	The ith document in the corpus
l_i	The length of the document d_i
w_{ij}	The jth token in the document d_i
z_{ij}	The latent topic of the token w_{ij}
n_{dz}	# of times that topic z is assigned to document d
n_{dw}	# of times that token w occurs document d
n_d	Total # of tokens in document d
n_{wz}	# of times that topic z is assigned to token w
n_z	# of times that topic z is assigned for the corpus

Table 1: Notations for LDA and CGS

CGS terminates, we group tokens by their *term frequency* (tf) and examine the topic distribution of tokens in each group. Let the group of tokens with $tf = m$ be denoted as G_m. I.e., $G_m = \{w_{ij} | i \in 1 \dots D,\ j \in 1 \dots l_i,\ n_{d_i w_{ij}} = m\}$, where n_{dw} represents the times that a token w occurs in a document d. The topic distribution of tokens in the token group G_m is the proportion of tokens that are assigned with $Z'(\leq Z)$ distinct topics, which is calculated as $r(Z', m) = \frac{\left|\left\{w_{ij} \middle| w_{ij} \in G_m,\ |\{z_{ij'} | j' \in 1 \dots l_i,\ w_{ij'} = w_{ij}\}| = Z'\right\}\right|}{|G_m|}$. We plot the topic distributions of tokens with different term frequencies for several m values in Fig. 3.

The first observation on these distributions is that multiple occurrences ($tf = m$) of the same token in a document are only assigned with a few distinct (much fewer than $\min(m, Z)$) topics. The second observation is that the distribution of the number of distinct topics is very skewed. In particular, we observe that (as shown in Fig. 3) the probability of assigning only one topic to multiple occurrences of a token is about 50%. These observations indicate that if a token w occurs n_{dw} times in a document d, it is highly probable (about 50% chance) that these n_{dw} tokens are assigned with one topic. We denote such observations as the property of ***skewed topic distributions***. Previous research works [5, 4, 7] utilize the sparsity in parameters n_{wz} and n_{dz} to design fast Gibbs sampling algorithms. This parameter-sparsity property denotes that as the sampling process proceeds most elements in n_{wz}, n_{dz} become zero. Our skewed topic distributions property is very different. We study the effect of the sparsity on the distributions of the number of distinct topics, and we discover that for the multiple occurrences of the same token the number of distinct topics

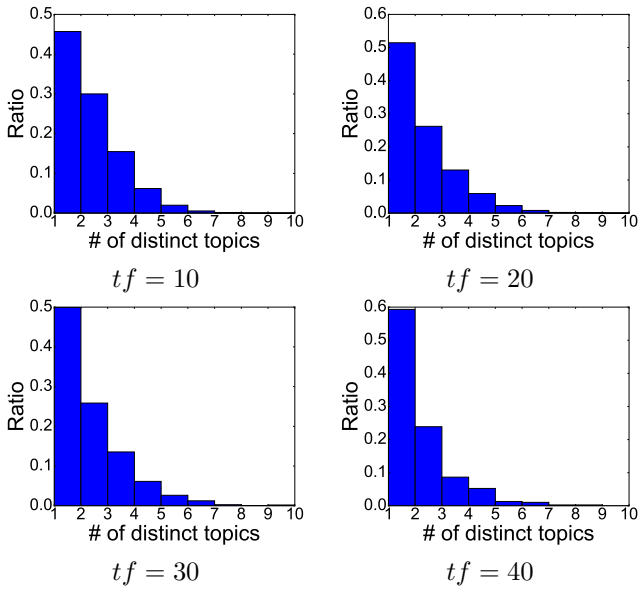


Figure 3: Topic distribution patterns learned by CGS for several token groups on NYTimes data set with $Z = 100$; $tf = X$ represents G_X

is usually very small.

The property of skewed topic distributions suggests that, when running CGS to learn LDA, it is unnecessary to sample each individual occurrence of a token. Instead, we can draw one topic and assign this topic to several occurrences of a token in one sampling step.

We propose the SGS strategy by utilizing the skewed topic distribution property in Fig. 4. The SGS strategy takes as input (a) a base Gibbs sampling algorithm for LDA (GS), which can be any existing LDA Gibbs Sampling algorithm, and (b) a group partition algorithm (GP), which is explained in Section 3.1.1. It works in two steps. The first step partitions the multiple occurrences of token w, whose term frequency is m, into s ($s \leq m$) groups $g_1, g_2, \ldots, g_s$ of size $m_1, m_2, \ldots, m_s$ ($\sum_{i=1}^{s} m_i = m$) by utilizing GP. The second step samples **one** representative topic for each group g_i with s_i occurrences of token w by calling GS. These two steps are repeated on all the distinct tokens in each document. This strategy can reduce the time complexity of sampling m occurrences of a token from $O(Zm)$ to $O(Zs)$ ($s < m$) if CGS is taken as the base GS. For other fast Gibbs sampling algorithms for LDA whose time complexity for sampling each token is $Z_f (< Z)$, applying SGS with the skewed topic distribution property on GS can reduce the time complexity of sampling m occurrences from $O(Z_f m)$ to $O(Z_f s)$. To differentiate a base Gibbs Sampling algorithm (GS) and a GS algorithm implementing the SGS strategy, we denote a base GS algorithm implementing the SGS strategy as an **SGS algorithm** in later discussions.

Please note that the behavior of an SGS algorithm is different from its corre-

Input: base Gibbs sampling algorithm for LDA (GS), group partition algorithm (GP)

1. For each iteration $i = 1, 2, \cdots$

(a) For each *distinct* token w in each document d

i. Partition m occurrences of w into groups $g_1, g_2, \ldots, g_s$ of size $m_1, m_2, \ldots, m_s$ ($\sum_{i=1}^{s} m_i = m$) utilizing GP

ii. For each group g_i for token w

A. Let z be the topic assignment to g_i in the previous iteration

B. Decrement n_{dz}, n_d, n_{wz}, and n_z by m_i

C. Sample z for g_i using GS

D. Increment n_{dz}, n_d, n_{wz}, n_z by m_i

Figure 4: SGS utilizing skewed topic distributions

sponding base GS. However, an SGS algorithm differs slightly from the behavior of the corresponding base GS algorithm (with bounded errors). The analysis of this bounded error is presented in Section 3.1.2.

3.1.1 Group Partition Algorithm

We design a uniform size group partition (USGP) algorithm. In USGP, a subsampling ration $q \in [0, 1.0]$ is defined to control the group size. Given q, the m occurrences of a token is partitioned into s groups. The group sizes are $\lceil mq \rceil, \lceil mq \rceil, \ldots, m - (s-1)\lceil mq \rceil$. If $\lceil mq \rceil < 1$, we put all m occurrences into one group. Given m, larger subsampling ratio q generates fewer groups with larger group size $\lceil mq \rceil$. Even USGP generates uniform size groups, the LDA model learned by SGS algorithms that utilize USGP still shows skewed topic distribution patterns. To verify this, we plot the topic distributions learned by the SGS algorithm with CGS as the base GS in Fig. 5. We can observe that the topic distributions of these tokens are still skewed. This is consistent with the trend of models learned by CGS in Fig. 3. We also observe that the SGS algorithm learns more skewed topic distributions than CGS. This is consistent with the intuition that group partitioning causes less number of distinct tokens to be sampled. Furthermore, the distribution for larger tf is more skewed. This is because the group size for smaller tf after group partitioning is smaller than the group size for larger tf. In the future, we will explore better group partition algorithms that can preserve the topic distributions.

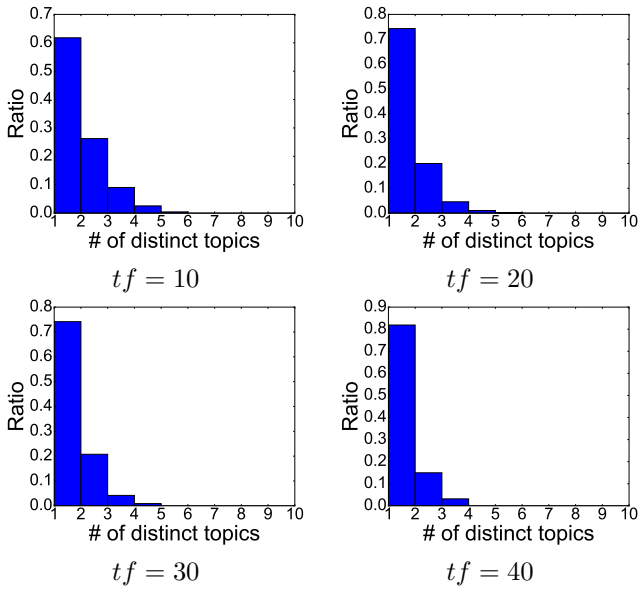


Figure 5: Topic distribution patterns learned by SGS algorithms (base GS = CGS, $q = 0.2$) on NYTimes data set with $Z = 100$

3.1.2 Error Bound Analysis

The sampling process of an SGS algorithm and its corresponding base GS algorithm (as shown in Fig. 2 and Fig. 4) are different. However, it has shown [11] that it is *intractable* to derive their accumulated difference through the *whole* sampling process because of the chain structure of MCMC algorithms. It is hard to reason the degree in which an SGS algorithm biases from GS at the *end* of the sampling process because, when sampling for each token, an SGS algorithm and its corresponding GS may assign different topics for the same token. The different topic assignments in one step may lead to different conditional probabilities in the remaining sampling iterations.

Nevertheless, to get a basic idea of the error that the subsampling brings, we analyze the difference between an SGS algorithm and its corresponding base GS in the sampling process for one token. We define δ_{dw} to be the difference between the conditional probabilities of an SGS algorithm and its corresponding GS that sample a token w in a document d. We are going to prove that starting from the same initial configuration, an SGS algorithm only biases from GS within a bounded error in one sampling step. Formally, we define $\delta_{dw} = ||p - p'||_1$, in which p' and p are conditional probabilities (Eq. (1)) of an SGS algorithm and its corresponding GS respectively. We prove that, given the same count matrices (n_{dz}, n_{wz}, and n_z), δ_{dw} has a limited upper bound which is proportional to the group size (which is linear in the subsampling ratio q).

We assume that before sampling for token w in document d, an SGS algo-

rithm and GS has the same count matrices (n_{dz}, n_{wz}, and n_z), and token w is assigned with topic $z = k_i$ in the previous iteration. In particular, we assume that the group on which the SGS algorithm is sampling has m_i tokens. Then the conditional probability of GS to sample different topics for a token w is shown as below:

$$p = \Bigg(\frac{n_{k_1 w} + \beta}{n_{k_1} + W\beta} \cdot \frac{n_{dk_1} + \alpha}{n_d + Z\alpha}, \ldots, \frac{n_{k_i w} + \beta - 1}{n_{k_i} + W\beta - 1} \cdot \frac{n_{dk_i} + \alpha - 1}{n_d + Z\alpha - 1}, \ldots, \frac{n_{k_Z w} + \beta}{n_{k_Z} + W\beta} \cdot \frac{n_{dk_Z} + \alpha}{n_d + Z\alpha} \Bigg)$$

The conditional probability of the SGS algorithm to sample different topics for a token w is

$$p' = \Bigg(\frac{n_{k_1 w} + \beta}{n_{k_1} + W\beta} \cdot \frac{n_{dk_1} + \alpha}{n_d + Z\alpha}, \ldots, \frac{n_{k_i w} + \beta - \mathbf{m_i}}{n_{k_i} + W\beta - \mathbf{m_i}} \cdot \frac{n_{dk_i} + \alpha - \mathbf{m_i}}{n_d + Z\alpha - \mathbf{m_i}}, \ldots, \frac{n_{k_Z w} + \beta}{n_{k_Z} + W\beta} \cdot \frac{n_{dk_Z} + \alpha}{n_d + Z\alpha} \Bigg)$$

p and p' are Z-dimensional vectors. p_i and p'_i represent probabilities (calculated using Eq. (1)), with which w is assigned topic k_i in GS and SGS respectively.

Before proceeding to analyze δ_{dw}, we introduce a theorem which is used in our proof.

Definition 1. *The Hilbert's projective metric between two vectors v, v' is defined as [12]:*

$$d(v, v') = \max_{i,j} (\log(\frac{v_i}{v'_i}) - \log(\frac{v_j}{v'_j})) \tag{2}$$

Theorem 1. *Let v and v' be positive vectors with unit sum, the L_1 norm of $v - v'$ is bounded by [12]:*

$$||v - v'||_1 = \sum_i |v_i - v'_i| \leq |1 - e^{\epsilon}| = \epsilon + O(\epsilon^2) \tag{3}$$

where $\epsilon = d(v, v')$.

Theorem 1 states that the sum of the absolute difference between v_i and v'_i is upper bounded by $\epsilon + O(\epsilon^2)$, in which $O(\epsilon^2)$ is a much small term comparing to ϵ and is less than $a\epsilon$ (a is a constant factor) because $\epsilon \leq 1$. Using Theorem 1, we can derive the upper bound of δ_{dw} as below:

Proof. δ_{dw} has a small upper bound

$$\begin{aligned}
&\delta_{dw} = ||p - p'||_1 \leq \epsilon + O(\epsilon^2) = \epsilon + a\epsilon^2 < (1+a)\epsilon \\
&= (1+a) \max_{i,j} (\log(\frac{p_i}{p'_i}) - \log(\frac{p_j}{p'_j})) \\
&= (1+a) \log \Big(\frac{n_{k_i w} + \beta - 1}{n_{k_i w} + \beta - m_i} \cdot \frac{n_{k_i} + W\beta - m_i}{n_{k_i} + W\beta - 1} \\
&\cdot \frac{n_{dk_i} + \alpha - 1}{n_{dk_i} + \alpha - m_i} \cdot \frac{n_d + Z\alpha - m_i}{n_d + Z\alpha - 1} \Big) \\
&= (1+a) \log \frac{C_3 C_4}{C_1 C_2} \\
&\text{where } C_1 = \frac{n_{k_i w} + \beta - m_i}{n_{k_i w} + \beta - 1}, \quad C_2 = \frac{n_{dk_i} + \alpha - m_i}{n_{dk_i} + \alpha - 1}, \\
&C_3 = \frac{n_{k_i} + W\beta - m_i}{n_{k_i} + W\beta - 1}, \quad C_4 = \frac{n_d + Z\alpha - m_i}{n_d + Z\alpha - 1} \\
&\text{Thus, } \delta_{dw} < (1+a) \log \frac{C_3 C_4}{C_1 C_2}
\end{aligned}$$

□

δ_{dw} is bounded by a small and positive upper bound $(1+a)\epsilon$ which is close to 0. We get this bound by analyzing the value range of the C factors using NYTimes data set as an example. Factors C_3 and C_4 are close to 1.0 because m_i is a very small number comparing to n_{k_i} (about $1M$ on average) and n_d (about 300 on average). C_1 and C_2 are also close to 1.0 because the averaged values of $n_{k_i w}$ and n_{dk_i} are around 200 and 40 respectively, which are also larger than m_i. Given the typical values of n_{k_i}, n_d, $n_{k_i w}$, and n_{dk_i}, we can get that $C_1 < C_3$ and $C_2 < C_4$. Thus, $\frac{C_3 C_4}{C_1 C_2} > 1$ and δ_{dw} is bounded by a small and positive value. On other data sets, the average values of these C factors scale corresponding to the size of the data set, and δ_{dw} is bounded in a similar manner.

We further explore how the subsampling ratio q affects the error bound of SGS algorithms by calculating the derivative of ϵ over m_i.

$$\begin{aligned}
\frac{\partial \epsilon}{\partial m_i} &= \underbrace{\frac{1}{n_{k_i w} + \beta - m_i} - \frac{1}{n_{k_i} + W\beta - m_i}}_{>0} \\
&+ \underbrace{\frac{1}{n_{dk_i} + \alpha - m_i} - \frac{1}{n_d + Z\alpha - m_i}}_{>0} \Rightarrow \frac{\partial \epsilon}{\partial m_i} > 0
\end{aligned}$$

A positive derivative shows that ϵ grows with the group size m_i which is linear in the subsampling ratio q as we discuss in Section 3.1.1. We verify this conclusion in Section 4.

3.2 SGS Utilizing Approximate Semantics

The SGS utilizing skewed topic distributions has a limitation in reducing N since this strategy creates one group for tokens that occur only once. To further reduce N we propose the SGS utilizing approximate semantics which is described in the remaining of this section.

The document-word representation of a corpus can be viewed as a bipartite graph. Fig. 6 shows an example of a small corpus which contains 4 documents and 7 distinct tokens. The edge from d_i to w_j means that document d_i contains word w_j. Let us use **semantics** to denote the set of distinct tokens (vocabulary) in the corpus. The idea of approximate semantics is inspired by the observation that a few documents can cover the overall semantics of a corpus. We use the data set NYTimes as an example to illustrate this idea. We first sort the documents in NYTimes by their number of distinct tokens in descending order. Let the number of distinct tokens in the first i documents be W_i. Recall that the total number of distinct tokens in the whole corpus is W. Then, we plot $\frac{W_i}{W}$ in Fig. 7(a). As we can see in Fig. 7(a), a small fraction of the documents (about 10,000 out of 300,000 $\approx 3\%$) covers the semantics of the whole NYTimes corpus. This observation indicates that the semantics of a corpus can be approximately represented by a small subset of documents.

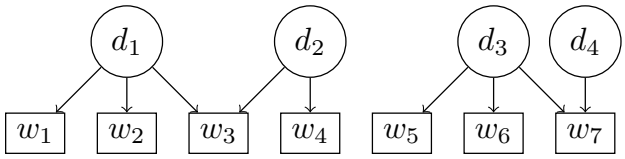
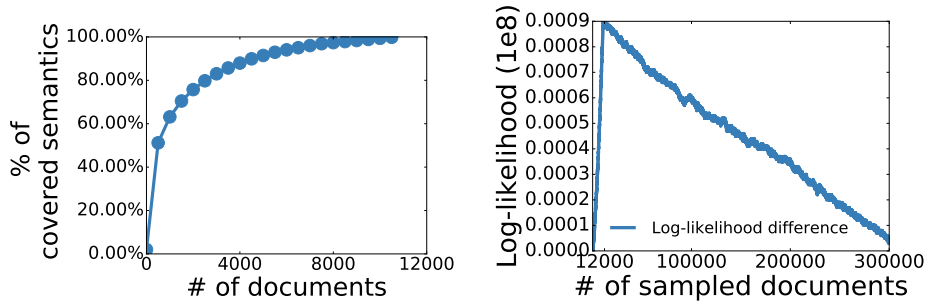


Figure 6: Bipartite representation of a small corpus



(a) (b)

Figure 7: Approximate semantics observations

A natural question following this observation is which subset of the corpus we should choose to approximately represent the semantics of the corpus. We

formulate the problem of choosing the representative subset of documents from a corpus as a set cover problem on a bipartite graph. The document-word representation of a corpus is formulated as a bipartite graph $G = (V, E)$. The graph has two types of nodes $V = \mathcal{D} \cup \mathcal{W}$, in which $\mathcal{D} = \{d_1, d_2, \cdots, d_D\}$ is the set of documents and $\mathcal{W} = \{w_1, w_2, \cdots, w_W\}$ is the set of distinct tokens. A graph edge $(d_i, w_j) \in E$ indicates that document d_i contains token w_j. To find a document subset to represent the semantics of the corpus, we adopt the greedy set cover algorithm [13] to find a subset of documents that contains all the distinct tokens in the corpus. The greedy set cover algorithm works in an iterative way. Initially the representative set is empty. In each iteration, the greedy algorithm first adds the document that contains the most distinct tokens into the representative set; then the added document and tokens contained in this document are removed from the graph. This process is repeated until all distinct tokens are contained in the representative set. Recall that the documents in the representative set are called covered-documents, which are used through the whole sampling process, and the remaining documents are called uncovered-documents, which are sampled in fewer iterations. For example, if we run the greedy set cover algorithm on the corpus shown in Fig. 6, the covered-documents are $\{d_1, d_3, d_2\}$.

To show that the covered-documents can approximately represent the semantics of a corpus, we designed two experimental tests to run CGS for one iteration on the NYTimes data set. In test-1, we randomly arrange the order of documents that are sampled. In test-2, we control the order of sampling by putting the covered-documents before the uncovered-documents. The differences between the log-likelihood of the LDA models learned by CGS with these two tests (values of test-2 minus the values of test-1) are shown in Fig. 7(b). We can observe that the differences grow faster on the first $12K$ documents, which is the approximate number of covered-documents. This means that the CGS that samples covered-documents first has a higher log-likelihood. It indicates that the covered-documents contain more information (semantics) of the corpus. This observation suggests that, during the sampling process of the whole corpus, we can focus on sampling the covered-documents first and reduce the number of iterations to sample uncovered-documents. If CGS runs on covered documents alone, the semantics will be lost too much compared with the results from the complete corpus. So the SGS strategy should still use the complete documents, but sample uncovered-documents for fewer times.

Fig. 7 also shows that, at the end of one iteration, the llh-value difference between the Gibbs sampling algorithms with different sampling orders is almost zero, which shows the validity of the proposed strategy. [14] proves that the sampling order affects the model convergence time and does not affect the correctness for some models although there is no general theoretical conclusion on whether changing the sampling order of tokens will affect the the correctness of Gibbs sampling algorithm for LDA, which will be an interesting future work.

The SGS strategy utilizing the approximate semantics property is shown in Fig. 8. For this strategy, we define a subsampling ratio $r \in [0, 1.0]$ to control the frequency that a GS algorithm runs on uncovered-documents. This strategy

Input: bipartite graph representation of a corpus $G = (V, E)$ in which $V = \mathcal{D} \cup \mathcal{W}$, base Gibbs sampling algorithm for LDA GS, subsampling ratio r

1. Use greedy set cover algorithm [13] to find a subset of documents S that covers the semantics of G
2. For each iteration $i = 1, 2, \cdots$
   (a) Run GS on S
   (b) If $i \mod (10 * r) = 0$, run GS on $\mathcal{D} - S$

Figure 8: SGS utilizing approximate semantics

works as follows. It first uses a greedy set cover algorithm to find the representative subset S. It then runs a base Gibbs sampling algorithm GS on S in each iteration. If the iteration number i satisfies that $i \mod (10 * r) = 0$, GS also runs on uncovered-documents $\mathcal{D} - S$. This means that GS runs on $\mathcal{D} - S$ every $10 * r$ iterations. In this way, the SGS algorithm skips sampling $\mathcal{D} - S$ in some iterations, which saves the running time significantly. We limit the subsampling ratio r in range $[0, 1.0]$ and set the consistent factor as 10 because out experiments show that when $10 * r > 10$ the effectiveness of SGS algorithms is unacceptable. Assuming that each document has L words on average and the base GS algorithm has complexity $O(Z_f)$ for sampling each token, then the complexity of the base GS algorithm is $O(DLZ_f)$ and the corresponding SGS algorithm has complexity $O((|S|+(D-|S|)*\frac{1}{10*r})LZ_f)$. Because the uncovered-documents, $\mathcal{D} - S$, are only sampled every $10 * r$ iterations and S is a small subset of $\mathcal{D}$, $|S| + (D - |S|) * \frac{1}{10*r} \ll D$, which means that the complexity of the SGS algorithm is much lower than the complexity of the base GS algorithm.

Data set	D	$\|S\|$	Running time (s)
NIPS	1,500	611	5
Enron	37,861	1,640	25
NYTimes	300,000	10,680	1,162
PubMed	8,200,000	21,095	37,447

Table 2: Set cover algorithm running time

The running time of the greedy set cover algorithm on four data sets is shown in TABLE 2. The running time of the greedy set cover algorithm on large data sets, NYTimes and PubMed, is about 20 minutes and 10 hours. This time seems long. However, it is approximately the same as the running time of one CGS iteration on the large data sets. Furthermore, the discovered representative set can be reused for any SGS algorithms with different parameter settings.

3.3 Discussions

We use ***sgs_GS***(q, r) to denote the SGS algorithm utilizing the skewed topic distribution property with subsampling ratio q and the approximate semantics property with subsampling ratio r. ***GS*** denotes the input base Gibbs sampling algorithm. In particular, ***sgs_GS***$(q, 0)$ denotes the base GS that only implements the skewed topic distribution property, ***sgs_GS***$(0, r)$ denotes the base GS that only implements the approximate semantics property. To achieve the maximum speedup with the skewed topic distribution property, we set $q = 1.0$ which means that we only sample once for m occurrences of a token. Similarly, to achieve the maximum speedup with the approximate semantics property, we set $r = 1.0$ which means that for every 10 iterations the uncovered-documents are sampled once. We limit $r \leq 1.0$ because higher r value results in bad effectiveness.

We demonstrate and verify the effect of the subsampling ratios q and r on the algorithm efficiency in Section 4. From the model learning perspective, the higher subsampling ratios q and r mean the fewer tokens and documents are sampled respectively. This reduces the effectiveness of the learned LDA model. The sampling error of the SGS strategy utilizing the skewed topic distribution property is proved in Section 3.1.2. The effect of subsampling ratios q and r on various effectiveness measures is presented and verified in Section 4. In real applications, it is a tradeoff to choose proper subsampling ratios q and r to have an expected improvement on efficiency and acceptable effectiveness. The suggestions on choosing p and r are summarized in the end of Section 4.

4 Experiments

We implement the proposed SGS strategy on four Gibbs sampling algorithms, *CGS* [2], *SparseLDA* [4], *AliasLDA* [5], and *F+Nomad LDA* [6]. We do not compare with any Gibbs samplings algorithms that are particularly designed for parallel or distributed environment (Section 5) because that is not the focus of this paper. *SparseLDA* and *F+Nomad LDA* are state-of-the-art fast Gibbs sampling algorithms for LDA. *AliasLDA* is not a Gibbs sampling algorithm, but is a more general Metropolis Hastings sampling algorithm for LDA. These four algorithms are used as the base Gibbs sampling algorithm (***GS***) to apply the SGS strategies in Fig. 4 and Fig. 8. They are denoted as ***cgs***, ***splda***, ***alias***, and ***flda*** respectively. The Gibbs sampling algorithms implementing the SGS strategy are denoted as ***sgs_cgs***, ***sgs_splda***, ***sgs_alias***, and ***sgs_flda*** respectively, which are called ***sgs_GS*** or SGS algorithms in later discussions. All the implementations are based on the open-source code in [6], and all the experiments are conducted on a workstation configured with Intel(R) Xeon(R) core (2.40GHz) and 256G RAM, running CentOS 6.5.

4.1 Data Sets

We conduct extensive experiments on four data sets, *NIPS, ENRON, NYTimes*, and *PubMed*, which are downloaded from UCI Bag of Words data collection [15]. The statistics of these data sets are shown in TABLE 3.

Data set	W	D	N
NIPS	12,419	1,500	746,316
ENRON	28,102	37,861	3,710,420
NYTimes	102,660	300,000	69,679,427
PubMed	141,043	8,200,000	483,450,157

Table 3: Statistics of data sets

4.2 Evaluation Measures and Parameter Settings

1. Log-likelihood (*llh*). Log-likelihood is widely utilized to measure the effectiveness of model inference algorithms (e.g., [16, 6, 7]). Note that confidence intervals on llh are hard to derive. We run the experiments with the same setting multiple times and get a range of llh to approximate the confidence intervals; however, the ranges are extremely small compared to the absolute llh values. Thus, we follow the convention in other works to report llh only. We use the likelihood defined in [16] which is shown as follows.

$$p(w, z|\alpha, \beta) = \Big(\prod_{i=1}^{D} \frac{\prod_{j=1}^{Z} \Gamma(\alpha + n_{d_i z_j})}{\Gamma(Z\alpha + n_{d_i})}\Big) \cdot \Big(\prod_{i=1}^{Z} \frac{\prod_{j=1}^{W} \Gamma(\beta + n_{w_j z_i})}{\Gamma(W\beta + n_{z_i})}\Big)$$

We define log-likelihood ratio (*llh-ratio*) to measure the effectiveness difference between ***GS*** and ***sgs_GS***. The *llh-ratio* of ***sgs_GS*** over ***GS*** is defined as

$$\textit{llh-ratio}(sgs\_GS, GS) = abs\Big(\frac{llh(sgs\_GS) - llh(GS)}{llh(GS)}\Big)$$

Smaller *llh-ratio* indicates ***sgs_GS*** and ***GS*** have similar *llh*.

2. KL-divergence. Log-likelihood only gives an overall estimation of the effectiveness of inference algorithms. It does not verify the quality of the learned topics. To further examine whether ***sgs_GS*** can learn similar topics as what ***GS*** learns, we utilize KL-divergence. Given that $\vec{\phi}_i = (\phi_{i1}, \phi_{i2}, \ldots, \phi_{iW})$ (a topic learned by ***GS***) and $\vec{\phi}'_j = (\phi'_{j1}, \phi'_{j2}, \ldots, \phi'_{jW})$ (a topic learned by ***sgs_GS***), the KL-divergence from $\vec{\phi}'_j$ to $\vec{\phi}_i$ is defined as $K_{i,j} = \sum_{w=1}^{W} \phi_{iw} \ln \frac{\phi_{iw}}{\phi'_{jw}}$.

3. Running time and speedup ratio. To measure how SGS strategy improves ***GS*** in efficiency, we record the average running time per iteration. We also calculate speedup ratios of ***sgs_GS*** and ***GS*** algorithms over ***cgs*** algorithm.

Algorithm	Top-2 words from 10 topics	Overlapping ratio in top-10 words
cgs	function, neuron, network, algorithm, image, object, system, cell, circuit, speech, learning, error, model, data, signal, unit	1.0
sgs_cgs$(0.2, 0)$	**function**, training, **neuron**, word, **network**, **algorithm**, **image**, **object**, **system**, **cell**, **circuit**, set, **learning**, input, **model**, **data**	0.82
sgs_cgs$(0.4, 0)$	**function**, set, point, **image**, **neuron**, noise, **learning**, **data**, recognition, training, **network**, **algorithm**, equation, **system**, **cell**, vector, **unit**, **model**	0.81
sgs_cgs$(0.6, 0)$	**function**, control, training, **object**, word, **network**, neural, **image**, **neuron**, **system**, **cell**, set, **learning**, input, **model**, **data**, recognition	0.80
sgs_cgs$(0.8, 0)$	**function**, control, training, set, **network**, **algorithm**, **image**, **neuron**, **system**, **cell**, **learning**, output, input, **model**, **data**, **unit**	0.79
sgs_cgs$(1.0, 0)$	**function**, control, set, **image**, **object**, **learning**, **data**, recognition, **network**, neural, pattern, presented, **system**, **cell**, memory, **model**, **neuron**	0.78

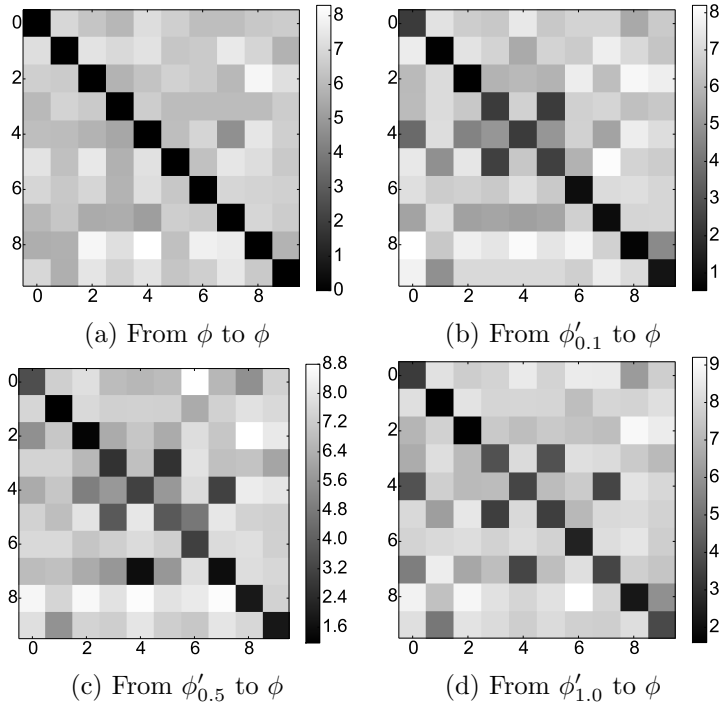
Table 4: Representative words of cgs and sgs_cgs on NIPS ($Z = 10$); overlapped words are highlighted

4. Parameter Settings. We set $\alpha = 50/Z$ and $\beta = 0.01$, which are also used in [6]. All ***sgs_GS*** and ***GS*** run 200/70/40 iterations on (NIPS, Enron)/NYTimes/PubMed data sets. Because ***cgs*** runs too slow on large data sets, fewer iterations are used on large data sets.

4.3 Effectiveness Analysis

In this section, we show that the proposed SGS algorithms can learn similar models as the corresponding base Gibbs sampling algorithms. We also examine the effect of subsampling ratios q and r on the effectiveness of SGS algorithms.

First, we examine the quality of the topics learned by SGS algorithms. Let the topics learned by ***cgs*** and ***sgs_cgs***$(q, 0)$ with $q = 0.1, 0.5$, and 1.0 be denoted as ϕ, $\phi'_{0.1}$, $\phi'_{0.5}$, and $\phi'_{1.0}$ respectively. Fig. 9 shows the *KL-divergence matrix* of



(a) From ϕ to ϕ (b) From $\phi'_{0.1}$ to ϕ

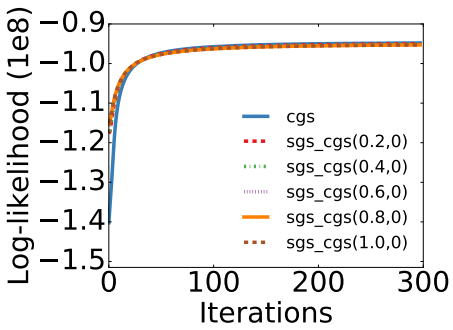
(c) From $\phi'_{0.5}$ to ϕ (d) From $\phi'_{1.0}$ to ϕ

Figure 9: KL-divergence matrix from ϕ' (learned by *sgs_cgs*) to ϕ (learned by *cgs*) on NIPS ($Z = 10$)

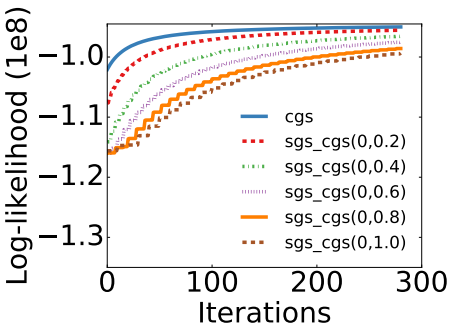
the NIPS data set. The ith row and jth column of the *KL-divergence matrix*[1] represents the KL-divergence from $\vec{\phi}'_j$ to $\vec{\phi}_i$. Dark pixels represent similar topic pairs, and light pixels represent dissimilar topic pairs. We observe that, as the subsampling ratio q grows, more dark pixels are off the diagonal. This trend denotes a worse matching from the learned topics of ***sgs_GS*** with higher subsampling ratio q to the topics learned through ***GS***. This trend is consistent with the analysis in Section 3.3. In TABLE 4 we show the representative words of the topics learned from the NIPS data set ($Z = 10$) using ***cgs*** and ***sgs_cgs***$(q, 0)$. The second column of TABLE 4 shows the top-2 words of the 10 topics learned by ***sgs_cgs*** and ***cgs***. The third column of TABLE 4 shows the ratio of overlapping words in the top-10 words for the 10 topics. When the subsampling ratio q increases, the overlapping words become less and the overlapping ratio decreases.

Second, we examine how the subsampling ratio q affects the effectiveness of SGS algorithms. Fig. 10(a) and Fig. 10(c) show the *llh* trends of ***sgs_cgs***$(q, 0)$ and ***cgs*** over iteration and time on the NYTimes data set, where q varies from 0.2 to 1.0. From Fig. 10(a) we can observe that the models learned by ***sgs_cgs***$(q, 0)$ and ***cgs*** have similar *llh* values at the end. Fig. 10(c) shows that ***sgs_cgs***$(q, 0)$

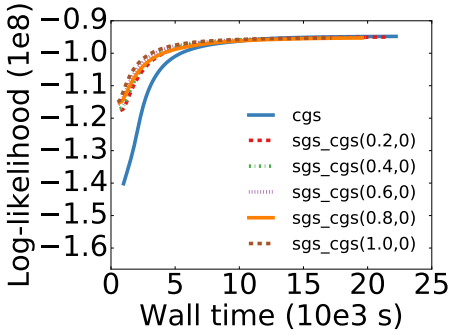
[1]Topic id is reordered to find a one-to-one match.



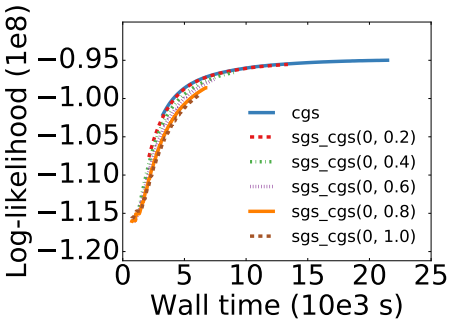
(a) Llh over iteration of ***sgs_cgs***$(q, 0)$



(b) Llh over iteration of ***sgs_cgs***$(0, r)$



(c) Llh over time of ***sgs_cgs***$(q, 0)$

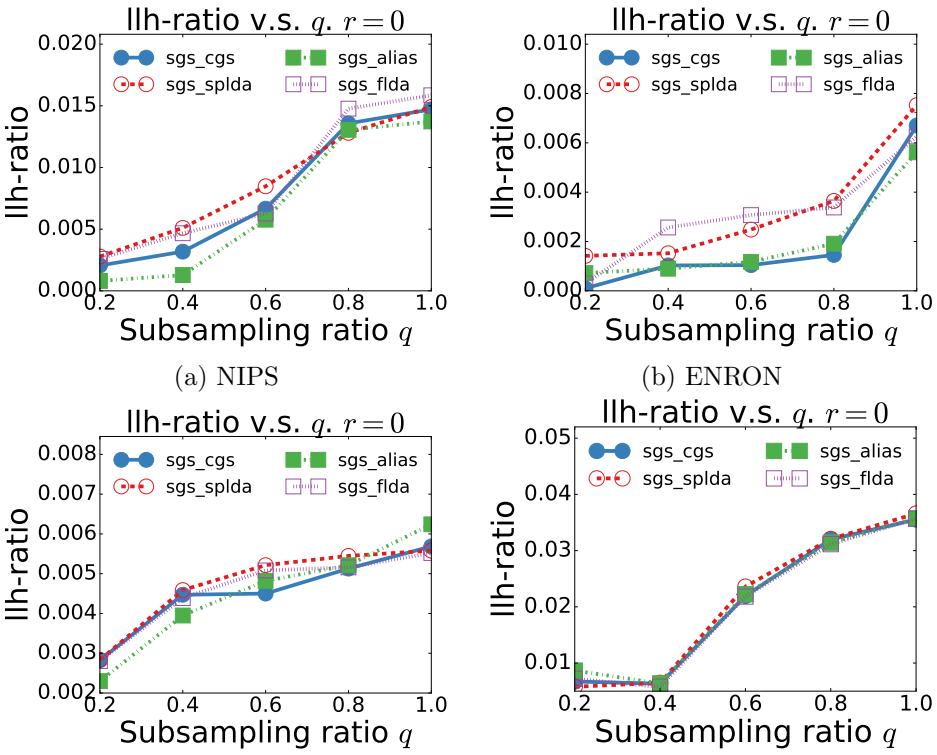


(d) Llh over time of ***sgs_cgs***$(0, r)$

Figure 10: Log-likelihood trend over iterations and wall time of ***sgs_cgs***(q, r) and ***cgs*** on NYTimes ($Z = 1000$)

can achieve higher *llh* than ***cgs*** within the same amount of time and ***sgs_cgs***$(q, 0)$ converges faster than ***cgs***. The ***sgs_GS*** algorithms with other base ***GS*** show similar patterns. This demonstrates that SGS algorithms can learn models that are similar to the models learned by the base ***GS*** algorithms and converge as the base ***GS*** algorithm. We further examine *llh-ratio*(***sgs_GS***, ***GS***) which is shown in Fig. 11 (with $Z = 1000$). We can observe that *llh-ratio*(***sgs_GS***, ***GS***) is higher when the subsampling ratio q is greater. This indicates that ***sgs_GS*** with higher subsampling ratio q learns a LDA model whose *llh* is more biased from that of ***GS***. We should note that the absolute *llh-ratio* of ***sgs_GS***$(q, 0)$ is still very small (less than 4%) which indicates that ***sgs_GS***$(q, 0)$ can achieve very similar *llh* as ***GS***. This is because fewer groups are sampled. This result is consistent the analysis in Section 3.3. In Fig. 12, we also plot the average δ_{dw} for all distinct tokens in all documents (the sampling error) which is derived in Section 3.1.2. It can be observed that greater subsampling ratio q results in larger δ_{dw}. Nevertheless, the averaged sampling error δ_{dw} is still relatively small (less than 0.08) on all the data sets.

Third, we examine how the subsampling ratio r affects the effectiveness of

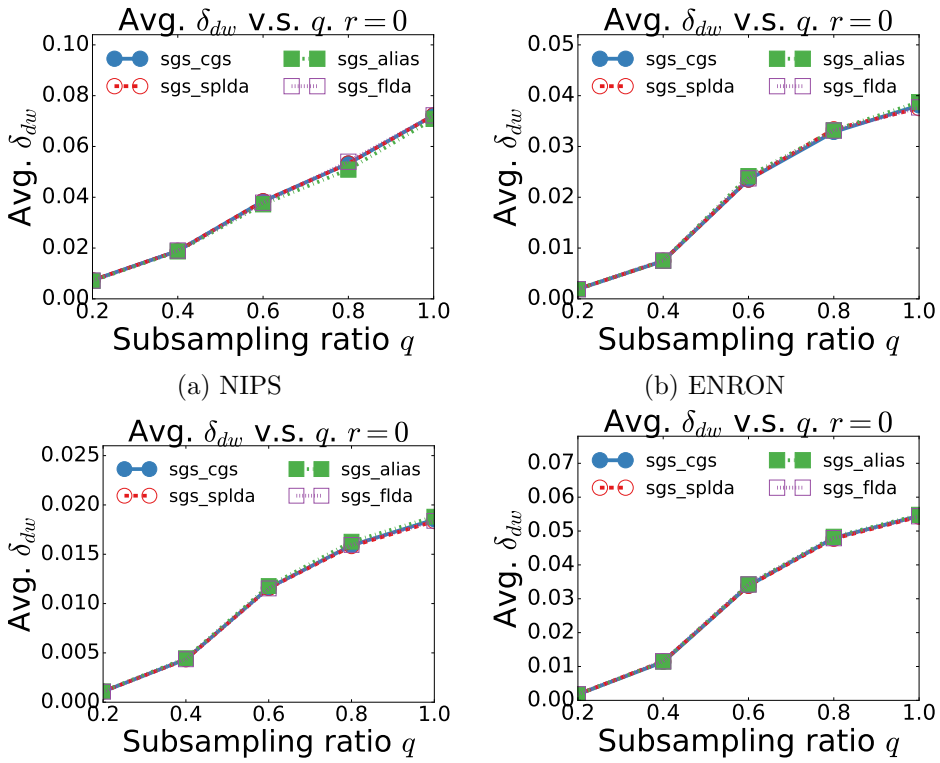


(a) NIPS (b) ENRON

(c) NYTimes (d) PubMed

Figure 11: llh-ratio v.s. subsampling ratio q ($Z = 1000$)

SGS algorithms. Fig. 10(b) and Fig. 10(d) show the *llh* trends of ***sgs_cgs***$(0, r)$ and ***cgs*** over iteration and time where r varies from 0.2 to 1.0. From Fig. 10(b) we can observe that ***sgs_cgs***$(0, r)$ with the lower subsampling ratio r learns models with the more similar *llh* as the models learned by ***cgs***. ***sgs_cgs***$(0, r)$ shows a zig-zag *llh* trend because ***cgs*** runs on uncovered-documents every $10 * r$ iterations in ***sgs_cgs***. Fig. 10(d) shows that ***sgs_cgs***$(0, r)$ achieve lower *llh* than ***cgs*** within the same amount of time and ***sgs_cgs***$(0, r)$ still converges as the ***cgs*** finally. We also observe that the final *llh* of ***sgs_cgs***$(0, r)$ with higher subsampling ratio r is biased more from ***cgs***. To study the effect of r on effectiveness of SGS algorithms, we present the *llh-ratio*(***sgs_GS***, ***GS***) in Fig. 13 with fixed $q = 1.0$ and various r. We can observe that higher r results in larger ratios. This result is consistent with our analysis: higher r indicates that ***SGS_GS*** runs fewer iterations on the uncovered-documents, which negatively affects the *llh* of the learned model. We note that on large data sets, high subsampling ratio r results in relatively unacceptable *llh-ratio* (≥ 0.1). We also observe that the subsampling ratio r has stronger effect on the *llh* than the subsampling ratio q. The reason is that SGS utilizing approximate semantics samples much fewer



(a) NIPS (b) ENRON

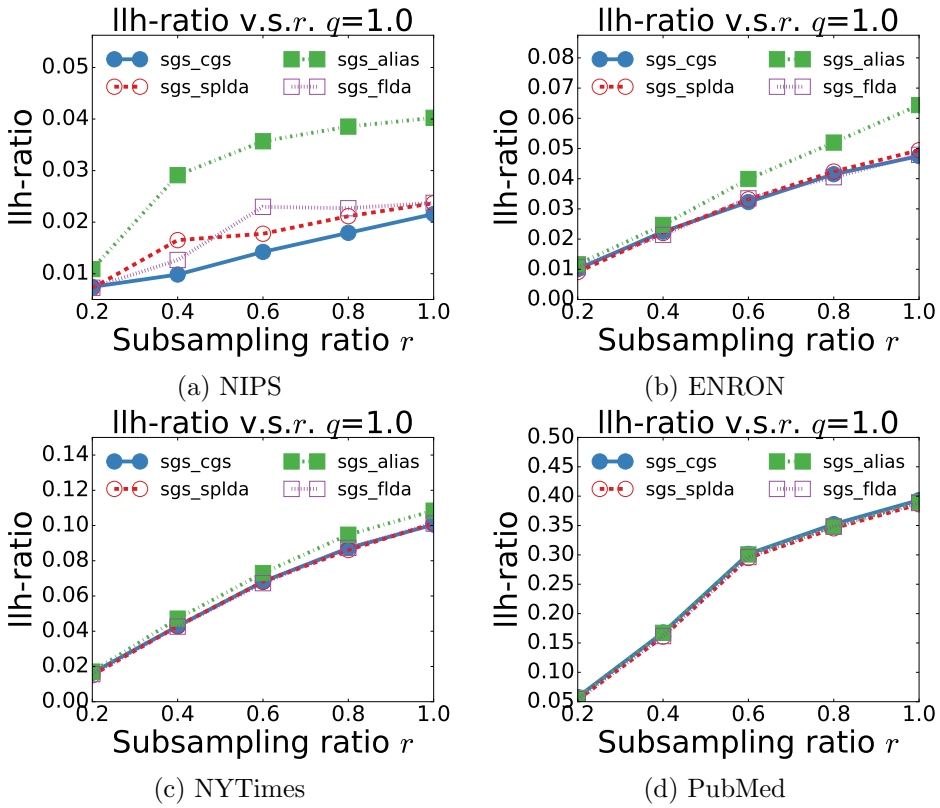
(c) NYTimes (d) PubMed

Figure 12: Average sampling error δ_{dw} v.s. subsampling ratio q ($Z = 1000$)

tokens (basically much fewer documents) than the SGS utilizing skewed topic distributions. The fewer sampled tokens indicate that the stronger r affects the results.

4.4 Efficiency Analysis

Fig. 14 compares the average running time per iteration of ***sgs_GS***(q, r) and ***GS*** on four data sets. We can observe that ***sgs_GS*** uses much less time than ***GS***. We investigate how the subsampling ratio q affects the efficiency of SGS algorithms. The speedup ratios of ***sgs_GS***$(q, 0)$ and ***GS*** over ***cgs*** are shown in Fig. 15. First, ***sgs_GS***$(q, 0)$ consistently has higher speedup ratio than the corresponding ***GS***. It means ***sgs_GS***$(q, 0)$ is faster than ***GS***. Second, when q grows, the speedup ratio also grows because higher q means that less groups (more tokens within one group) are sampled in every iteration. Note that the trend of ***alias*** and ***splda*** is consistent with the experimental results in [6] because we use the implementation of in [6]. We also note that this trend is different from [5]; without the specific implementations of [5], it is hard to repeat their results.



(a) NIPS (b) ENRON

(c) NYTimes (d) PubMed

Figure 13: llh-ratio v.s. subsampling ratio r ($Z = 1000$)

We also examine how the subsampling ratio r affects the efficiency of SGS algorithms. The speedup ratios of ***GS*** and ***sgs_GS***$(1.0, r)$ over ***cgs*** are shown in Fig. 16. The figure shows that ***sgs_GS***$(1.0, r)$ consistently has higher speedup ratio than the corresponding ***GS***. In addition, when r grows, the speedup ratio of ***sgs_GS***$(1.0, r)$ also grows. These are consistent with our analysis in Section 3.3: a higher r means that uncovered-documents are sampled in fewer iterations.

Finally, we check the efficiency of ***sgs_GS*** under different model complexities Z. We vary Z and fix the subsampling ratios $q = 1.0$ and r to be 0 and 0.3, which is a balanced choice between effectiveness and efficiency. On NIPS, Enron, and NYTimes data sets, Z varies from 2000 to 10000. On the PubMed data set, Z varies from 1000 to 5000 due to the limitation of RAM size. The speedup ratios of ***GS*** and ***sgs_GS*** over ***cgs*** are shown in TABLE 5. On all data sets, the speedup ratios of ***splda***, ***alias***, and ***flda*** increase when Z increases. This is consistent with the design of these methods [5, 4, 6]. Overall the ***sgs_GS*** algorithms significantly improves the corresponding ***GS*** algorithms. On small data sets (NIPS and Enron), ***sgs_GS***$(1.0, 0)$ is about twice faster than the corresponding ***GS***, and ***sgs_GS***$(1.0, 0.3)$ is $3 \sim 4$ times faster than the corresponding

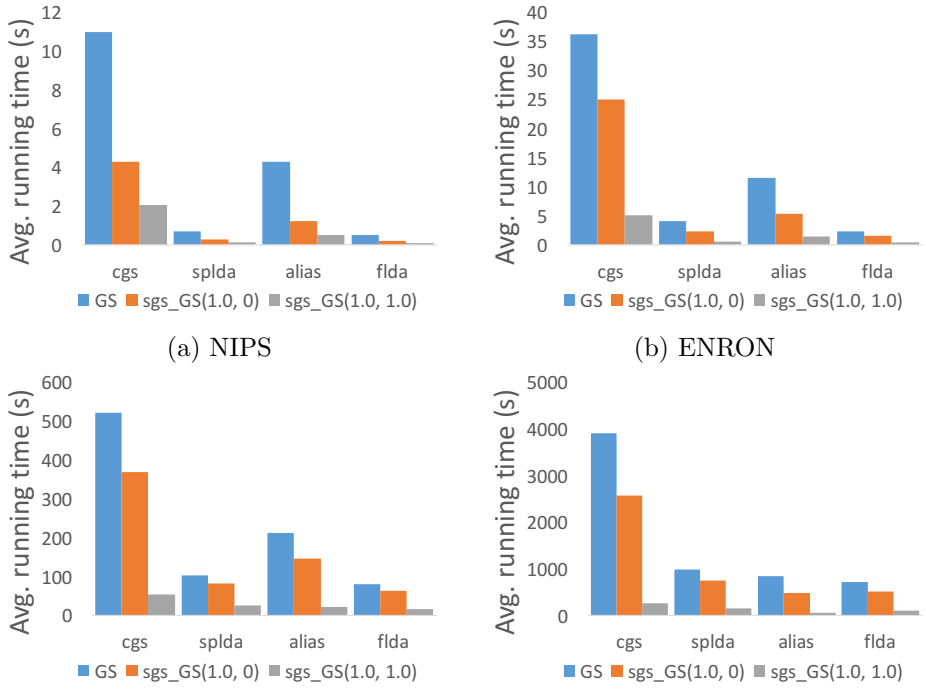
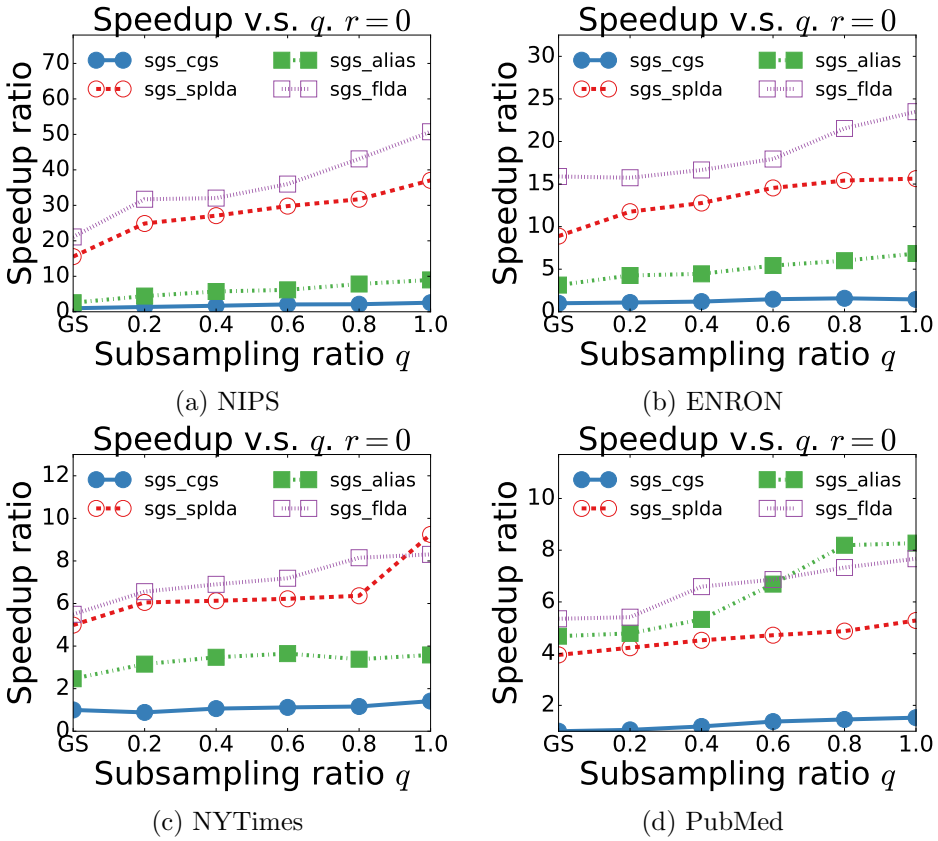


Figure 14: Comparison of average running time per iteration ($Z = 1000$)

GS. On large data sets (NYTimes and PubMed), the improvement on running time is also significant. ***sgs_splda***$(1.0, 0)$ and ***sgs_flda***$(1.0, 0)$ are 50% faster than ***splda*** and ***flda*** respectively. ***sgs_alias***$(1.0, 0)$ is twice faster than ***alias*** for most parameter settings. ***sgs_splda***(1.0, 0.3) and ***sgs_flda***(1.0, 0.3) are about $2 \sim 3$ times faster than ***splda*** and ***flda*** respectively. ***sgs_alias***$(1.0, 0.3)$ is about $2 \sim 5$ times faster than ***alias*** in different parameter settings. An impressive example is that, ***cgs*** takes 5 hours to run one iteration on PubMed with $Z = 5000$; with the same setting, ***sgs_cgs***$(1.0, 0.3)$ takes only 0.5 hours.

Summary: After demonstrating how subsampling ratios affect the effectiveness and efficiency, we would like to mention that, in real applications, it is a tradeoff to choose proper subsampling ratios q and r to have expected speedup and acceptable effectiveness. Since the subsampling ratio q does not affect effectiveness too much as shown in Fig. 11, we suggest to choose $q = 1.0$ to have a maximum speedup with little sacrifice on effectiveness. Since large subsampling ratio r results in unacceptable effectiveness, we recommend to choose a relatively small r value depending on the data set size.



(a) NIPS (b) ENRON

(c) NYTimes (d) PubMed

Figure 15: Comparison of speedup of ***sgs_GS*** methods v.s. subsampling ratio q ($Z = 1000$)

5 Related Work

We already review single-thread fast Gibbs sampling algorithm in Section 1. In this section we review *parallel and distributed* Gibbs sampling algorithms. Real world text corpora are massive, which brings challenges to the scalability of topic model inference. Newman et al. [17] introduce *AD-LDA* algorithm, which is the first attempt to distribute the Gibbs sampling algorithm to infer topic models. *AD-LDA* partitions the documents into disjoint groups and assign partitions to different processors on which Gibbs sampling is performed. Ihler et al. [11] improve [17] by designing a better partition strategies which impose a constraint that different groups do not share words. [11] also proves the error bound of the distributed Gibbs sampling algorithm. Smola et al. [16] have designed *Yahoo LDA* which builds a fault tolerant and scalable inference system for topic models. Yuan et al. [7] propose *LightLDA* that first observes the access patterns of the parameters of topic models. By caching frequent accessed

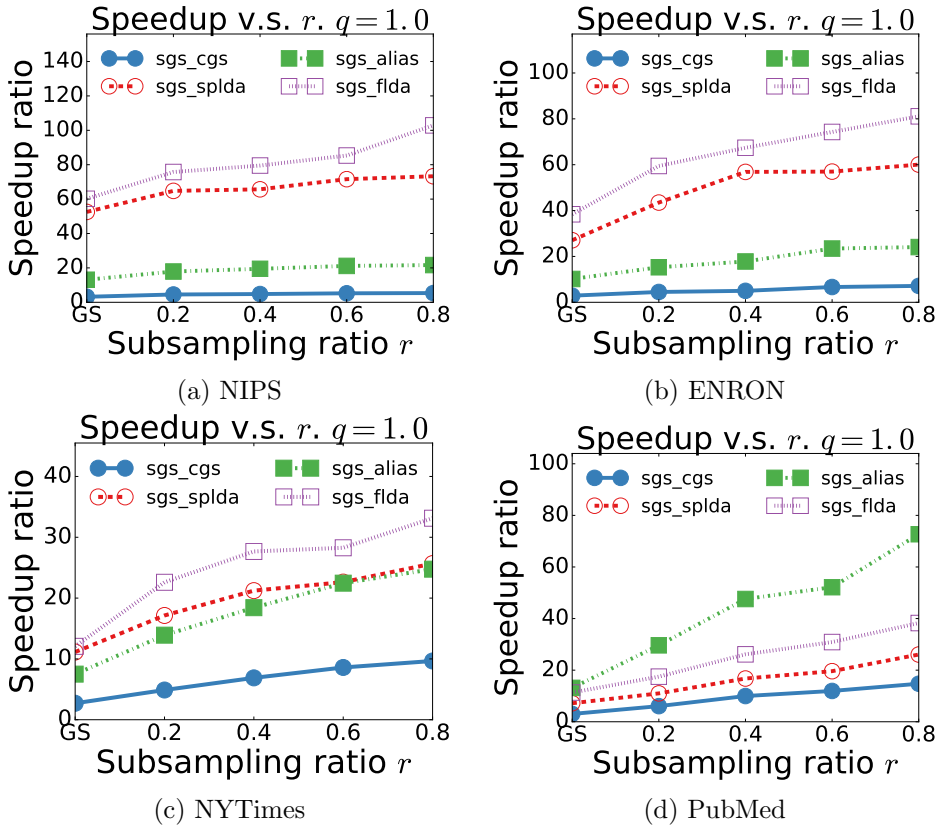
Algorithms	$Z=$ 2000	$Z=$ 5000	$Z=$ 8000	$Z=$ 10000	$Z=$ 2000	$Z=$ 5000	$Z=$ 8000	$Z=$ 10000
	NIPS				Enron			
sgs_cgs$(1.0, 0)$	2.61	3.00	2.67	2.54	1.71	1.62	1.72	1.74
sgs_cgs$(1.0, 0.3)$	4.89	4.68	4.74	4.10	3.63	3.92	3.89	3.93
splda	23.56	33.90	40.66	48.57	14.75	14.76	17.03	19.46
sgs_splda$(1.0, 0)$	50.87	67.66	74.34	84.45	20.66	20.68	23.13	25.68
sgs_splda$(1.0, 0.3)$	84.50	143.44	124.05	134.91	42.13	47.84	53.72	60.24
alias	5.55	12.37	17.93	29.82	6.24	11.53	20.75	24.63
sgs_alias$(1.0, 0)$	19.03	41.00	60.69	74.82	12.73	26.73	41.91	48.44
sgs_alias$(1.0, 0.3)$	30.30	64.19	98.53	117.34	27.38	54.86	79.90	106.21
flda	33.36	52.77	67.02	82.82	18.17	30.66	33.39	34.00
sgs_flda$(1.0, 0)$	76.58	149.99	204.64	286.78	33.19	35.65	46.62	56.84
sgs_flda$(1.0, 0.3)$	141.40	244.71	325.33	423.65	71.44	98.90	104.25	134.32
Algorithms	$Z=$ 2000	$Z=$ 5000	$Z=$ 8000	$Z=$ 10000	$Z=$ 2000	$Z=$ 5000	$Z=$ 8000	$Z=$ 10000
	NYTimes				PubMed			
sgs_cgs$(1.0, 0)$	1.43	1.43	1.42	1.44	1.52	1.54	1.57	
sgs_cgs$(1.0, 0.3)$	4.23	4.22	4.23	4.27	14.74	15.10	13.74	
splda	5.02	7.35	8.94	8.47	3.95	3.08	3.49	
sgs_splda$(1.0, 0)$	8.02	9.47	10.15	12.53	5.28	4.23	5.50	
sgs_splda$(1.0, 0.3)$	16.41	19.53	28.68	29.27	11.00	12.48	13.90	
alias	4.34	10.21	13.62	18.86	4.68	7.96	13.41	
sgs_alias$(1.0, 0)$	7.06	14.24	20.88	32.13	8.27	11.45	23.80	
sgs_alias$(1.0, 0.3)$	17.15	50.77	75.22	113.48	29.63	46.91	93.02	
flda	7.37	8.31	8.98	10.29	5.41	3.73	4.63	
sgs_flda$(1.0, 0)$	8.51	11.51	13.31	14.95	7.67	5.92	7.79	
sgs_flda$(1.0, 0.3)$	18.07	29.83	36.48	29.91	17.48	15.67	19.43	

Table 5: Speedup of sgs_GS over cgs varying Z

data and adopting parameter server framework [18], *LightLDA* improves the efficiency and scalability of topic models. Yu et al. [6] propose *F+Nomad LDA* which carefully partitions data into disjoint groups and designs an asynchronous framework to process the disjoint groups in parallel. Chen et al. [19] propose *WarpLDA*, which is a cache efficient $O(1)$ algorithm, for LDA by designing Monte-Carlo Expectation Maximization (MCEM) inference algorithm.

6 Conclusions and Future Work

In this paper, we propose a novel and general strategy Sub-Gibbs Sampling (SGS), to improve the efficiency of any Gibbs sampling algorithms for LDA. The SGS strategy utilizes two properties that we observed in text corpora, the tokens in a document have skewed topic distributions and the semantics of a corpus can be approximately covered by a subset of documents in this corpus. We theoretically prove that the error of SGS algorithm is bounded by a small upper bound. We implemented the SGS strategy on the traditional Collapsed Gibbs Sampling (*CGS*) algorithm and three state-of-the-art Gibbs sampling algorithms (*FastLDA*, *AliasLDA*, and *F+Nomad LDA*). The experimental results on four real data sets showed that the SGS algorithms can learn similar models as those of other Gibbs sampling algorithms with much better efficiency. In particular the proposed strategy is $2 \sim 100$ times faster than CGS and $2 \sim 5$ times faster than *FastLDA*, *AliasLDA*, and *F+Nomad LDA* algorithms. In the future we



(a) NIPS (b) ENRON

(c) NYTimes (d) PubMed

Figure 16: Comparison of speedup of ***sgs_GS*** methods v.s. subsampling ratio r ($Z = 1000$)

will explore better group partition algorithms and find a close form error bound for the SGS strategy.

Acknowledgment

This research work is supported by NSF #1633330 and #1345232. All the experiments are conducted on NMSU's bigdat cluster[2] which is supported by CNS-1337884.

References

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *JMLR*, vol. 3, pp. 993–1022, 2003.

[2] `http://bigdat.nmsu.edu/`

[2] 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.

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

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

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

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

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

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

[9] P. M. Fenwick, "A new data structure for cumulative frequency tables," *Softw., Pract. Exper.*, vol. 24, no. 3, pp. 327–336, 1994.

[10] G. Heinrich, "Parameter estimation for text analysis," Technical Note, University of Leipzig, Germany., 2008.

[11] A. T. Ihler and D. Newman, "Understanding errors in approximate distributed latent dirichlet allocation," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 5, pp. 952–960, 2012.

[12] P. J. Bushell, "Hilbert's metric and positive contraction mappings in a banach space," *Arch. Rational Mech. Anal*, 1973.

[13] C. Lund and M. Yannakakis, "On the hardness of approximating minimization problems," *J. ACM*, vol. 41, no. 5, pp. 960–981, Sep. 1994.

[14] B. D. He, C. D. Sa, I. Mitliagkas, and C. Ré, "Scan order in gibbs sampling: Models in which it matters and bounds on how much," in *NIPS 2016, December 5-10, 2016, Barcelona, Spain*, 2016, pp. 1–9. [Online]. Available: http://papers.nips.cc/paper/6589-scan-order-in-gibbs-sampling-models-in-which-it-matters-and-bounds-on-how-much

[15] "UCI bag of words data sets," https://archive.ics.uci.edu/ml/datasets/Bag+of+Words.

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

[17] D. Newman, A. U. Asuncion, P. Smyth, and M. Welling, "Distributed algorithms for topic models," *JMLR*, vol. 10, pp. 1801–1828, 2009. [Online]. Available: http://doi.acm.org/10.1145/1577069.1755845

[18] Q. Ho, J. Cipar, H. Cui, S. Lee, J. K. Kim, P. B. Gibbons, G. A. Gibson, G. R. Ganger, and E. P. Xing, "More effective distributed ML via a stale synchronous parallel parameter server," in *Neural Information Processing Systems*, 2013, pp. 1223–1231. [Online]. Available: http://papers.nips.cc/paper/4894-more-effective-distributed-ml-via-a-stale-synchronous-parallel-parameter-server

[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, 2016. [Online]. Available: http://www.vldb.org/pvldb/vol9/p744-chen.pdf