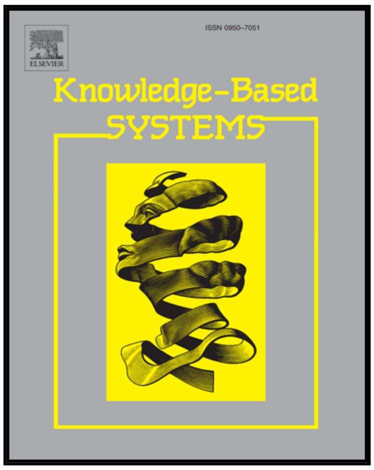


FastBTM: Reducing the Sampling Time for Biterm Topic Model

Xingwei He, Hua Xu, Jia Li, Liu He, LinLin Yu

PII: S0950-7051(17)30278-2
DOI: 10.1016/j.knosys.2017.06.005
Reference: KNOSYS 3932

To appear in: *Knowledge-Based Systems*

Received date: 23 December 2016
Revised date: 31 May 2017
Accepted date: 3 June 2017

Please cite this article as: Xingwei He, Hua Xu, Jia Li, Liu He, LinLin Yu, FastBTM: Reducing the Sampling Time for Biterm Topic Model, *Knowledge-Based Systems* (2017), doi: 10.1016/j.knosys.2017.06.005

Highlights

- We combine alias method, Metropolis-Hastings and factorized strategy and propose an acceleration algorithm of BTM, FastBTM.
- FastBTM reduces sampling complexity of biterm topic model from O(K) to O(1).
- FastBTM converges faster than BTM without degrading topic quality.
- FastBTM is effective for both short text datasets and long document datasets.

FastBTM: Reducing the Sampling Time for Biterm Topic Model

Xingwei He[a], Hua Xu[a,*], Jia Li[a], Liu He[b], LinLin Yu[c]

[a]*State Key Laboratory of Intelligent Technology and Systems, Tsinghua National Laboratory for Information Science and Technology, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China*
[b]*Beijing University of Posts and Telecommunications, Beijing 100876, China*
[c]*Shanghai Jiao Tong University, Shanghai 200240, China*

Abstract

Due to the popularity of social networks, such as microblogs and Twitter, a vast amount of short text data is created every day. Much recent research in short text becomes increasingly significant, such as topic inference for short text. Biterm topic model (BTM) benefits from the word co-occurrence patterns of the corpus, which makes it perform better than conventional topic models in uncovering latent semantic relevance for short text. However, BTM resorts to Gibbs sampling to infer topics, which is very time consuming, especially for large-scale datasets or when the number of topics is extremely large. It requires $O(K)$ operations per sample for K topics, where K denotes the number of topics in the corpus. In this paper, we propose an acceleration algorithm of BTM, FastBTM, using an efficient sampling method for BTM, which converges much faster than BTM without degrading topic quality. FastBTM is based on Metropolis-Hastings and alias method, both of which have been widely adopted in Latent Dirichlet Allocation (LDA) model and achieved outstanding speedup. Our FastBTM can effectively reduce the sampling complexity of biterm topic model from $O(K)$ to $O(1)$ amortized time. We carry out a number of experiments on three datasets including two short text datasets, Tweets2011 Col-

*Corresponding author
Email addresses: hexingwei852@sina.com (Xingwei He), xuhua@tsinghua.edu.cn (Hua Xu)

Preprint submitted to Elsevier *June 5, 2017*

lection dataset and Yahoo! Answers dataset, and one long document dataset, Enron dataset. Our experimental results show that when the number of topics K increases, the gap in running time speed between FastBTM and BTM gets especially larger. In addition, our FastBTM is effective for both short text datasets and long document datasets.

Keywords: BTM, Topic Model, Alias Method, Metropolis-Hastings, Acceleration Algorithm

---

1. Introduction

With the popularity of social networks, such as Twitter and microblogs, detecting the latent topic of short text is crucial for many natural language processing areas, such as implicit feature detection [1], social recommendation systems [2, 3, 4], question retrieval in community question answering [5], sentiment analysis [6, 7] and so on. Instance messages are often very short. For example, a tweet message can contain 140 characters at most. Hence, it's obvious that such short texts contain poorer information than traditional normal documents (e.g. research papers and news articles), which makes it very difficult to model topic for short documents.

Conventional topic models, such as PLSA [8] and LDA [9] infer topics by capturing word co-occurrence patterns [10] implicitly. However, due to the poor information of short texts, word co-occurrence patterns are very sparse in such texts. If we directly apply conventional topic models on short documents, these conventional topic models will suffer from word co-occurrence pattern sparsity severely. To tackle the sparsity problem, Yan et al. [11, 12] proposed biterm topic model (BTM), a novel topic model. They assume that a biterm is an unordered word-pair co-occurring in a short text and the two words in the biterm own the same hidden topics. Different from the conventional topic models, BTM models the topic from the whole biterm corpus, rather than the document. By learning from the aggregated co-occurrence patterns, BTM alleviates the sparsity problem at document-level, leading to the conclusion that BTM outperforms LDA and PLSA at short text topic modeling.

Because of its highly effectiveness for short text topic modeling, BTM shows promising in many areas. For example, Wang et al. (2014) [13] employed BTM for extracting keywords. Xia et al. (2015) [14] modified BTM (d-BTM) for headline-based social news clustering.

In this general context, it is necessary to speed up the sampling process of BTM so that BTM will be suitable for large datasets, as well as benefitting on-line topic detection. Recently, acceleration algorithm of LDA has achieved great success, such as FastLDA [15], SparseLDA [16], AliasLDA [17], LightLDA [18], WarpLDA [19]. AliasLDA adopts alias method [20, 21] and Metropolis-Hastings [22, 23, 24, 25, 26] to reduce the time complexity to $O(K_d)$, where K_d represents the number of actually instantiated topics in document. Furthermore, LightLDA uses factorized strategy: it utilizes two proposals instead of one, and alternates between them for inferring topic. BTM uses Gibbs sampling method [27, 28, 29] for inferring topic. Gibbs sampling is very time consuming, costing $O(K)$ operations per sample for K topics, where K denotes the number of topics in the corpus.

Inspired by AliasLDA and LightLDA, we propose a highly efficient model FastBTM, the core of which includes alias method, Metropolis-Hastings and factorized strategy. Compared with Gibbs sampling method, our FastBTM reduces the time complexity from $O(K)$ to $O(1)$.

We carry out extensive experiments on three different datasets, Tweets2011 Collection dataset, Yahoo! Answers dataset and Enron dataset. We use the log likelihood as a metric to test whether our FastBTM will degrade topic quality. Then, we compare our FastBTM with BTM w.r.t. convergence speed on the three datasets to verify the effectiveness of our proposed model.

The main contributions of our work are as follows:

(1) We propose an acceleration algorithm of BTM, FastBTM, effectively reducing sampling complexity of biterm topic model from $O(K)$ to $O(1)$.

(2) We conduct extensive experiments to compare our FastBTM with BTM, and the results show that our algorithm converges faster than BTM without

degrading topic quality.

(3) We carry out experiments on both short text datasets and long document dataset and our experimental results demonstrate that our model is effective for both short text datasets and long document datasets.

The rest of the paper is organized as follows. In Sect.2, we introduce the related work. In Sect.3, we give a brief introduction of biterm topic model and the Gibbs sampling method for BTM. We present our FastBTM model in Sect.4 and conduct experiments to show the effectiveness of our FastBTM in Sect.5. In Sect.6, we conclude our work.

2. Related Work

In this section, we will briefly review the related work about the topic models on short text and the acceleration models for LDA.

Topic model on short text: In the last decade, topic models on long documents, such as news articles and academic papers, have achieved great success. Many topic models are proposed such as latent semantic analysis (LSA) [30], probabilistic latent semantic analysis (PLSA) [8] and Latent Dirichlet Allocation (LDA) [9], among which LDA has proven to be most promising for topic mining on such long documents. However, conventional topic models, like LDA, suffer from severe data sparsity in short texts like, Twitter. Therefore, many researchers have turned to the study of topic inferring for short texts recently. Ramage et al. (2009) [31] proposed Labeled LDA and gave a scalable implementation (2010) [32], which enables explicit models of text content associated with replies, hashtags, emoticons, and the like. Then Ramage et al. (2011) [33] proposed Partially Labeled Dirichlet Allocation (PLDA) and they assume that only topics associated with the documents labels can be used by the document. Hong et al. (2010) used different aggregation strategies, such as training the model on aggregated user profiles, training the model on aggregated term profiles, to train a standard topic model and author-topic model [34] in short text environments effectively. Zhao et al. (2011) [35] developed Twitter-LDA model, where they posit that a tweet only contains a single topic. It is worth mentioning that Yan et al. (2013) [11] proposed biterm topic model (BTM), which can capture topic for short texts by modeling the generation of word co-occurrence patterns in the whole corpus. For BTM can infer topic for short texts effectively, BTM has achieved great success and been widely used in recent years. Xu et al. (2013) [36] used BTM for semantic similarity matching for short texts. Chu et al. (2014) [37] employed BTM for web service orchestration topic mining. Inspired by LDA, Pan et al. (2014) [38] proposed biterm-based Dirichlet process for BTM. Yan et al. (2015) [39] extended the biterm topic model for bursty topic discovery in microblogs. Chen et al. (2015) [40] proposed Twitter-BTM, user based aggregation for BTM. And Li et al. (2016) [41] proposed User-IBTM for online hashtag suggestion.

Acceleration model for LDA: LDA is a successful topic model which has been widely used in sentiment analysis and text mining. In the last decade, many efficient and scalable models have been proposed for LDA. For example, Porteous et al. (2008) [15] proposed FastLDA, which takes equivalent samples but costs significantly less than K operations on average. Yao et al. (2009) [16] decomposed the collapsed sampler to utilize the sparse structures of LDA and proposed SparseLDA, which can reduce the sampling complexity of LDA from $O(K)$ to $O(K_d + K_w)$, where K_d represents the number of actually instantiated topics in document d and K_w denotes the number of topics assigned for word w. Compared with FastLDA, SparseLDA not only speeds up the Gibbs sampling but also reduces memory usage. However, with the number of documents increases, K_d gets larger and at last equals approximately to K resulting in the vanishing of the word sparsity. To solve this issue, Li et al (2014) [17] combined alias method and Metropolis-Hastings and proposed AliasLDA, which only utilizes the document sparsity to reduce the sampling complexity to $O(K_d)$. Furthermore, Yuan et al. (2015) [18] decomposed the Gibbs sampling equation into two terms, which are used as doc proposal and word proposal. They sampled new topic from corpus proposal, word proposal in turn and proposed LightLDA reducing the sampling complexity to $O(1)$. Chen et al (2016)

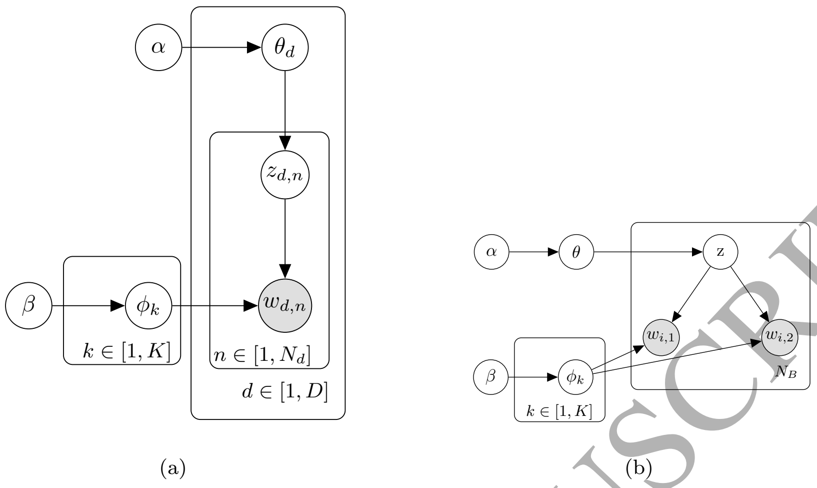


Figure 1: Graphical representation of (a) LDA, (b) BTM.

[19] designed WarpLDA to optimize the cache hit rate.

However, BTM uses Gibbs sampling method to model topic for document, which takes $O(K)$ operations per sample and is very time consuming, especially for large scale datasets and large topic number K. For BTM is very important in topic modeling for short text, it is necessary to speed up the sampling process. Inspired by AliasLDA and LightLDA, we propose a highly efficient model FastBTM, which successfully reduces the time complexity from $O(K)$ to $O(1)$.

3. Biterm topic model

Our work optimizes the biterm topic model, mainly using Metropolis-Hastings and alias method. So we will give a brief introduction of biterm topic model and the Gibbs sampling method for BTM.

3.1. LDA

LDA describes each document d as a multinomial distribution over topics θ_d and each topic as a multinomial distribution over words. We give the generative process of LDA as follows:

1. Draw a topic distribution from dirichlet distribution with hyperparameter α.

$$\theta_d \sim Dir(\alpha) \tag{1}$$

2. Draw a word distribution from the dirichlet distribution with hyperparameter β for topic t.

$$\phi_t \sim Dir(\beta) \tag{2}$$

4. Draw a topic z from the multinomial distribution θ_d.

$$z \sim Multi(\theta_d) \tag{3}$$

5. Draw a word w from the multinomial distribution ϕ_z.

$$w \sim Multi(\phi_z) \tag{4}$$

LDA's graphical representation is shown in Fig. 1(a).

3.2. Biterm Topic Model

However, if directly applying LDA for short text, we will suffer from sparse word co-occurrence problem. To solve the sparse word co-occurrence challenge of short text, Yan et al. [11, 12] proposed biterm topic model (BTM). They posit that the more frequently two words appear in the fixed window, the more likely they are to share the same topic. Based on this idea, they construct biterms, namely two words, from a fixed window. In addition, they assume that the whole corpus rather than each document is associated with a multinomial distribution over topics. The generative process of BTM is shown in Fig. 1(b). The following is the generative process of BTM:

1. Transform the whole corpus C into biterm set B.

2. Draw a topic distribution θ from a Dirichlet distribution with hyperparameter α for the whole biterm set B

$$\theta \sim Dir(\alpha) \tag{5}$$

3. Draw a word distribution ϕ_t from a Dirichlet distribution with hyperparameter β for each topic t

$$\phi_t \sim Dir(\beta) \tag{6}$$

4. Draw a topic z from the multinomial θ for a biterm b

$$z \sim Multi(\theta) \tag{7}$$

5. Draw two words from the multinomial ϕ_z for a biterm b

$$w_i, w_j \sim Multi(\phi_z) \tag{8}$$

BTM leverages the word co-occurrence patterns of the whole corpus instead of one document and generates two words, sharing the same topic, per sample. So BTM makes good use of corpus-level co-ccurring words, and performs well at short text topic inferring.

3.3. Gibbs Sampling Method for BTM

Similar to LDA [42], BTM employs Gibbs sampling to infer topic approximately. Compared with variational inference and maximum posterior estimation, Gibbs sampling is more simple and efficient, which makes it become a widely applicable Markov chain Monte Carlo algorithm. We directly give the conditional probability in Equation (9) and more details can be found in [11, 12].

$$P(z|\mathbf{z}_{-b}, B, \alpha, \beta) \propto (n_z^{-b} + \alpha)\frac{(n_{w_i|z}^{-b} + \beta)(n_{w_j|z}^{-b} + \beta)}{(\sum_{w=1}^{V} n_{w|z}^{-b} + V\beta)^2} \tag{9}$$

where B denotes the whole biterm set and $b = (w_i, w_j)$. $\mathbf{z}_{-b}$ indicates the topic assignments for all biterms, where biterm b is excluded. Both α and β are hyperparameter for Dirichlet distribution. n_z^{-b} denotes the number of biterm assigned to topic z excluding b. $n_{w_i|z}^{-b}$ and $n_{w_j|z}^{-b}$ are the number of word w_i and w_j assigned to topic z except biterm b, respectively. V is vocabulary size of the corpus.

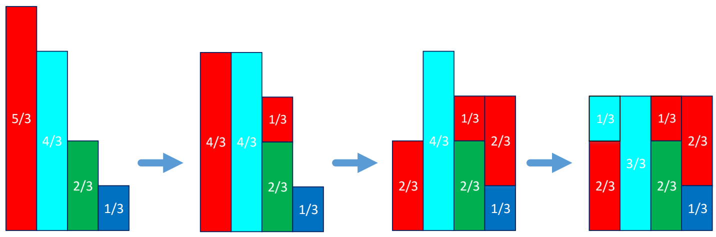


Figure 2: Illustration of how to build an alias table with discrete probabilities 5/12, 1/3, 1/6, 1/12. We scale all of these probabilities by 4 so that a probability 1/4 would have height 1. More details about generating the alias table can be found in Algorithm 1.

4. FastBTM

In this section, we will give a detail description of our FastBTM.

4.1. Alias Method

Alias method is a highly efficient algorithm for sampling from a discrete probability distribution. Given n probabilities $p_1, p_2 ... p_n$, if we use a general method, it will take $O(n)$ operations to generate a sample. However, if drawing from a uniform distribution, it only requires $O(1)$ operations. Inspired by the uniform distribution, alias method creates an alias table and simulates uniform sampling. Even though, creating alias table will take $O(n)$ operations. However, if we take a sampling for n times, the sampling can be finished in $O(1)$ amortized time. Li et al. (2014)[17] describe the algorithm of alias method. Different from Li, we adopt a modified version Vose's Alias Method [43], which is numerically stable and practical. We show how to build an alias table using discrete probability distribution in Fig. 2. Algorithm 1 and Algorithm 2 illustrate this method.

4.2. Metropolis-Hastings

Inspired by LightLDA [18], we employ a factorized strategy, so that Metropolis-Hastings algorithm is cheap and has high acceptance probability. The difference between LDA and BTM is that the conditional distribution of BTM contains three parts instead of two. So we decompose the conditional distribution of

Algorithm 1 Generate AliasTable

Input: a set of n discrete probabilities $p_1, p_2...p_n$

Output: *AliasTable* and *ProbTable*

Create *AliasTable* and *ProbTable* , each of size n
Create two list, *SmallList* and *LargeList*
Set $p_i = n \times p_i, i \in (1, 2...n)$
for each $i \in (1, 2...n)$ **do**
    if $p_i < 1$ **then**
        SmallList add i
    else
        LargeList add i
    end if
end for
while *SmallList* and *LargeList* are not empty **do**
    $l = SmallList.pop(1)$, $g = LargeList.pop(1)$
    $ProbTable[l] = p_l$
    $AliasTable[l] = g$
    $p_g = p_g + p_l - 1$
    if $p_g < 1$ **then**
        SmallList add g
    else
        LargeList add g
    end if
end while
while *LargeList* is not empty **do**
    $g = LargeList.pop(1)$
    $ProbTable[g] = 1$
end while
while *SmallList* is not empty **do**
    $l = SmallList.pop(1)$
    $ProbTable[l] = 1$
end while

Algorithm 2 Sampling Process

Input: *AliasTable* and *ProbTable*

Output: the sampling integer

```
r=randint(n)
f=random(0,1)
if f < ProbTable[r] then
    return r
else
    return AliasTable[r]
end if
```

BTM into three parts: $(n_z+\alpha)$, $\frac{(n_{w_i|z}+\beta)}{(\sum_w^V n_{w|z}+V\beta)}$ and $\frac{(n_{w_j|z}+\beta)}{(\sum_{w=1}^V n_{w|z}+V\beta)}$. To improve the mixing rate, we choose all of the three parts as the proposal distributions. If only one part is used as the proposal distribution, the acceptance probability of state transition will be very low, under which condition the Markov chain Monte Carlo can't converge to the global optimal states. We call $(n_z+\alpha)$ as corpus proposal and $\frac{(n_{w_i|z}+\beta)}{(\sum_{w=1}^V n_{w|z}+V\beta)}$ as word proposal.

4.2.1. word proposal

We define p_{w_i} as the word proposal distribution.

$$p_{w_i}(z) \propto \frac{(n_{w_i|z}+\beta)}{(\sum_{w=1}^V n_{w|z}+V\beta)} \tag{10}$$

When state s translates to state t, the acceptance probability is $min(1,\pi_{w_i})$, where π_{w_i} is

$$\begin{aligned}\pi_{w_i} = & \frac{(n_{w_i|t}^{-b}+\beta)(n_{w_j|t}^{-b}+\beta)}{(n_{w_i|s}^{-b}+\beta)(n_{w_j|s}^{-b}+\beta)}\frac{(\sum_{w=1}^V n_{w|s}^{-b}+V\beta)^2}{(\sum_{w=1}^V n_{w|t}^{-b}+V\beta)^2} \\ & \cdot\frac{(n_t^{-b}+\alpha)}{(n_s^{-b}+\alpha)}\frac{(n_{w_i|s}+\beta)}{(n_{w_i|t}+\beta)}\frac{(\sum_{w=1}^V n_{w|t}+V\beta)}{(\sum_{w=1}^V n_{w|s}+V\beta)}\end{aligned} \tag{11}$$

We use p_{w_i} as the proposal distribution for BTM. If we directly sample topic from word proposal distribution, it will take $O(K)$ operations per sample and it isn't cheaper than Gibbs sampling. To sample from p_{w_i} in $O(1)$, we employ alias table, like AliasLDA [17] and LightLDA [18]. From Algorithm 1, we know that it will spend $O(K)$ time to construct the alias table for p_{w_i}. Algorithm 2 shows that once the alias table is created, sampling from the alias table can be finished in $O(1)$ time. In addition, after finishing sampling, we only take $O(1)$ operations to compute the acceptance probability π_{w_i}. In fact, $n_{w|z}$ changes slowly over time. For example, only two counters for old topic are reduced by one and two counters for new topic are added by one per sample. Based on this idea, there is no need to update the alias table every sample. So we can repeatedly draw from p_{w_i} for $O(K)$ times using the same alias table costing $O(K)$ time. Finally, sampling from p_{w_i} can be accomplished in $O(1)$ amortized time per sample.

4.2.2. corpus proposal

We define p_c as the corpus proposal distribution.

$$p_c(z) \propto (n_z + \alpha) \tag{12}$$

When state s translates to state t, the acceptance probability is $min(1, \pi_c)$, where π_c is

$$\begin{aligned}\pi_c = &\frac{(n_t^{-b} + \alpha)}{(n_s^{-b} + \alpha)} \frac{(n_{w_i|t}^{-b} + \beta)(n_{w_j|t}^{-b} + \beta)}{(n_{w_i|s}^{-b} + \beta)(n_{w_j|s}^{-b} + \beta)} \\ &\cdot \frac{(\sum_w^V n_{w|s}^{-b} + V\beta)^2}{(\sum_w^V n_{w|t}^{-b} + V\beta)^2} \frac{(n_s + \alpha)}{(n_t + \alpha)}\end{aligned} \tag{13}$$

In light of the above, we have shown that we can finish sampling from word proposal in $O(1)$ time. Similarly, we can use the method sampling from corpus proposal in $O(1)$ amortized time per sample like word proposal. However, it's not necessary to construct the alias table for corpus proposal. Like LightLDA, we decompose $p_c(z)$ into two parts: n_z and α. The first term n_z denotes how many biterms are assigned to topic z. We use C_i to store the topic assigned for the i-th biterm b_i and use N_B to denote the length of the biterm set of the corpus.

$$n_z = \sum_{i=1}^{N_B} [C_i = z] \tag{14}$$

Furthermore, we employ C_i as the alias table for corpus proposal. We will prove that sampling from C_i equals to sampling from n_z. Given biterm b_i and topic k, we give the probability of drawing a topic for biterm b_i from n_z in Equation 15 and give the probability drawing a topic from C_i in Equation 16, where Z_i denotes the topic assignment for b_i.

$$p(Z_i = k) = \frac{n_k}{N_B} \tag{15}$$

$$p(Z_i = k) = \frac{\sum_{i=1}^{N_B} [C_i = k]}{N_B} \tag{16}$$

From Equation 14, we know $n_k = \sum_{i=1}^{N_B} [C_i = k]$. So we can conclude that sampling from C_i equals to sampling from n_z. We substitute C_i, uniform distribution, for n_z. In this way, we don't have to construct an alias table for n_z, so we can draw a topic from C_i in $O(1)$ time.

In general, we use symmetric Dirichlet priors α for BTM. So the second term α is a constant for all biterms, namely uniform distribution. Therefore, we can draw a topic from the second term in $O(1)$ time.

To summarize, both C_i and α are uniform distribution. So we can draw from the corpus proposal in constant time. What's more, we also save a lot of space, since we don't have to build an alias table. We illustrate how to infer a topic from the corpus proposal without constructing an alias table in Fig. 3.

4.3. Combining Proposals for FastBTM

In the last subsection, we resolve the equation of BTM's Gibbs sampling into three parts, two word proposals and one corpus proposal. Each of them can be employed as the proposal distribution for MH algorithm. But if we only employ one proposal, we can't achieve a better mixing rate because each of them is far from the true conditional probability of BTM. To improve the

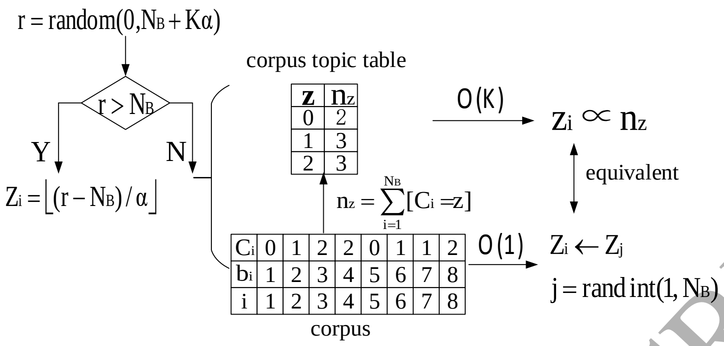


Figure 3: Illustration of how to infer a topic from the corpus proposal without needing construct an alias table.

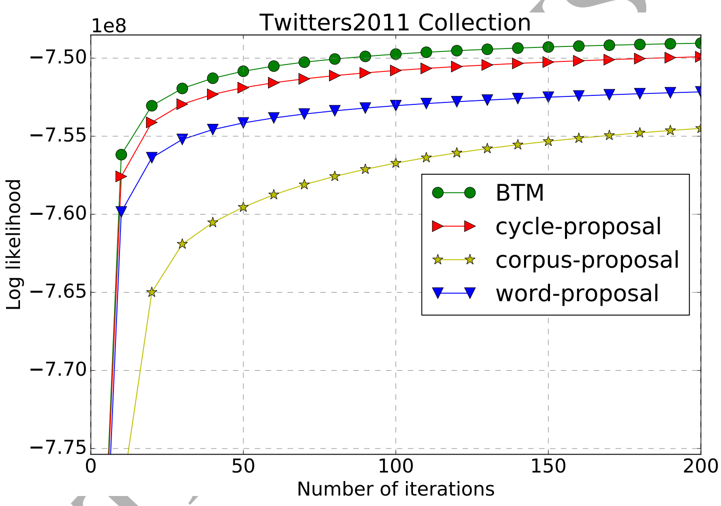


Figure 4: Performance of different FastBTM Metropolis-Hastings proposals, on the Tweets2011 Collection dataset with K = 200 topics

mixing rate, we infer new topic from cycle proposal, employed by LightLDA [18]. For $p_b(z) \propto p_c(z)p_{w_i}(z)p_{w_j}(z)$, we draw new topic from corpus proposal, word proposal p_{w_i} and word proposal p_{w_j} in turn. Fig. 4 illustrates the convergence performance of different strategies with the change of iteration, from which we conclude that corpus proposal performs worst in all strategies and word proposal performs much better than corpus proposal. Moreover, cycle proposal's performance is far better than only using one proposal and very close to BTM's

Algorithm 3 FastBTM

Input: word proposal, corpus proposal

Output: new topic t

Use word proposal to create alias tables $A[V]$ for every word of the corpus
for each $i \in (1, 2...MH\_step)$ **do**
  Draw topic t from corpus proposal
  Compute p, the acceptance probability of state transition $s \to t$
  r=random(0,1)
  if $r < p$ **then**
    $s = t$
  end if
  Draw new topic t from word proposal p_{w_i}
  Compute p, the acceptance probability of state transition $s \to t$
  r=random(0,1)
  if $r < p$ **then**
    $s = t$
  end if
  Draw new topic t from word proposal p_{w_j}
  Compute p, the acceptance probability of state transition $s \to t$
  r=random(0,1)
  if $r < p$ **then**
    $s = t$
  end if
  Update $A[w_i]$ and $A[w_j]$ as needed
end for

proving that using cycle proposal we can achieve a better mixing rate. Hence, our strategy is drawing new topic from corpus proposal, word proposal p_{w_i} and word proposal p_{w_j} in turn. Algorithm 3 shows the brief process of FastBTM.

5. EXPERIMENTAL RESULTS

In this section, we will show the effectiveness of our FastBTM. We carry out our experiments on three different datasets including two short text datasets and one long document dataset and we compare the convergence speed of our FastBTM with BTM.

5.1. Environment and Datasets

Yan et al. (2013) provide open-source implementation of BTM via C++[1] for us. We implement our FastBTM and BTM using java code. Moreover, we perform all experiments on a Linux server with 32GB memory and Intel(R) Core(TM) i7-4790 CPU with 4.00GHz clock rate. The operation system of our Linux server is Ubuntu 14.04 64bit.

We conduct our experiments on three different datasets: Tweets2011 Collection, Yahoo! Answers and Enron, which are different in biterm set size, vocabulary size and document length. Both Tweets2011 Collection and Yahoo! Answers are short text datasets. To verify the effectiveness of our model on long documents, we also perform on Enron Email Dataset, which is a long document dataset compared to the previous two. We provide all kinds of statistics from these datasets in Table 1. The three datasets are as follows:

1. **Tweets2011 Collection**[2] is a short text dataset, which contains approximately 16 million tweets crawled from www.twitter.com between January 23rd and February 8th, 2011. Yan et al. (2013) [11] use this dataset to verify the effectiveness of BTM on short texts.
2. **Yahoo! Answers**[3] is a famous Community-based Question Answering (CQA) system. We use the L4 - Yahoo! Answers Manner Questions, version 1.0 dataset provided by the Yahoo Webscope Program[4]. This dataset contains 142,627 question-answer pairs. Because the answers are very long, we remove

[1]http://code.google.com/p/btm/
[2]http://trec.nist.gov/data/tweets/
[3]http://answers.yahoo.com/
[4]http://webscope.sandbox.yahoo.com/

Table 1: Datasets and their statistics. V is the vocabulary size. L denotes the total number of words in the dataset. D is the number of documents and N_B is the total number of biterms of the dataset. L/V is the average number of a word. L/D is the average lenght of a document.

DATASET	V	L	D	N_B	L/V	L/D
Tweets2011	71651	11217660	1688627	45914027	156	6
Yahoo! Answers	45948	1502136	142627	8155678	32	10
Enron	28102	3710420	39861	85594204	132	93

the answers and only use the questions as a short text dataset, like Tweets2011 Collection.

3. **Enron**[5] Email Dataset is a long document dataset and contains data from about 150 users. This dataset is used by Li et al. (2014) [17] to show the speedup of AliasLDA.

For all datasets, we remove the stop words and remove the words containing non-Latin characters. What's more, we also convert capital letters into lower case. At last, we remove duplicate tweets and documents.

5.2. Evaluation Metrics

We utilize the likelihood of the whole corpus as our evaluation metric, which is widely used by conventional topic models like LDA to verify the convergence. Given the parameters Θ and Φ, we first compute the probability of biterm b_i in Equation (17). Then, we show how to compute the likelihood of the whole corpus of BTM in Equation (18).

[5]orig source: http://www.cs.cmu.edu/enron

$$P(b_i|\Theta, \Phi) = \sum_{k=1}^{K} \theta_k \phi_{k,w_{i,1}} \phi_{k,w_{i,2}} \tag{17}$$

$$P(\mathbf{B}|\Theta, \Phi) = \prod_{i=1}^{N_B} \sum_{k=1}^{K} \theta_k \phi_{k,w_{i,1}} \phi_{k,w_{i,2}} \tag{18}$$

where K denotes the number of topics for the corpus. Θ is a K-dimensional multinomial distribution and θ_k indicates the probability of topic k. Φ is a $K * V$ matrix, ϕ_k is a V-dimentional multinomial distribution and $\phi_{k,w}$ denotes the probability of word w conditioned on topic k. We directly give the computational process in Equation (19) and Equation (20).

$$\phi_{k,w} = \frac{n_{w|k} + \beta}{\sum_{w=1}^{V} n_{w|k} + V\beta} \tag{19}$$

$$\theta_k = \frac{n_k + \alpha}{N_B + K\alpha} \tag{20}$$

To further evaluate the quality of the topics inferred, we also resort to point-wise mutual information (PMI) [44, 45], which is a popular metric to measure the topic coherence. To evaluate on the Tweets2011 dataset, we extract the co-occurring word pair (w_i, w_j) in a sliding window of 5-words from about 4M English Wikipedia articles.

We compute the PMI-Score of a topic k as follows:

$$PMI(k) = \frac{1}{T(T-1)} \sum_{1 \le i < j \le T} PMI(w_i, w_j) \tag{21}$$

$$PMI(w_i, w_j) = log \frac{p(w_i, w_j)}{p(w_i)p(w_j)} \tag{22}$$

where T denotes the number of the most probable words of topic k.

5.3. Baseline and Experimental Setup

In this paper, we use the original BTM proposed by Yan et al. (2013) [11] as our baseline. BTM uses Gibbs sampling to infer new topic, while our FastBTM resort to alias method and MH algorithm. To demonstrate the effectiveness of our model, we compare our FastBTM with BTM w.r.t. convergence speed on the three datasets.

We set $\alpha = 0.1$ and $\beta = 0.01$ for both BTM and FastBTM in all experiments. To achieve a satisfactory acceptance rate, we set the number of Metropolis-Hasting sampling steps equals to 2. To illustrate that our FastBTM converges faster than BTM, we perform our experiments on different topic number, setting $K = 200$, 500 and 1000 and using a single computational thread. In all experiments, we run each model for 200 iterations and compute the log likelihood after every 10 iterations. In the meantime, we calculate the average time elapsed for per Gibbs sampling iteration.

5.4. Experimental Results and Analysis

In this subsection, we carry out experiments on two short text datasets to demonstrate that our FastBTM converges faster than BTM. In addition, we also conduct experiments on a long document dataset to illustrate that our FastBTM is suitable not only for short text datasets, but also for long document datasets. For all datasets, we mainly compare our FastBTM with BTM in three aspects: (a) First, we want to show that our FastBTM doesn't degrade topic quality, namely the clustering result of our model is as good as BTM. So we use log likelihood as a function of iterations; (b) Next, we want to prove that our FastBTM converges faster than BTM. Hence, we use log likelihood as a function of running time; (c) To further demonstrate the effectiveness of our FastBTM, we compare our model against BTM w.r.t. the runtime of per iteration.

5.4.1. Analysis on short text datasets

We first compare our FastBTM with BTM on two short text datasets. Fig. 5 illustrates log likelihood as a function of iterations on the three datasets. The second row and the third row are the convergence results on Tweets2011 Collection and Yahoo! Answers, respectively. And the first column, the second column and the third column correspond to the results of $K = 200$, $K = 500$ and $K = 1,000$, respectively. From the second row, we can see that whatever the topic number K is 200, 500, or 1000, our FastBTM converges to almost the same log likelihood as BTM on Tweets2011 Collection. Similarly, we can observe

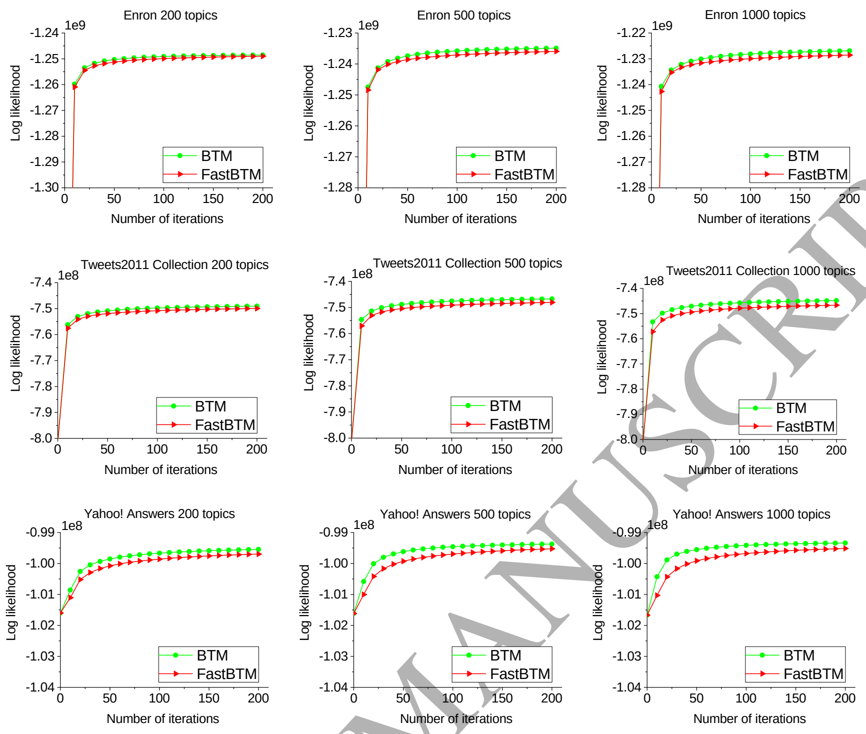


Figure 5: Log likelihood as a function of number of iterations for BTM and FastBTM

that our FastBTM performs as good as BTM on Yahoo! Answers dataset. So we can draw a conclusion that our acceleration algorithm doesn't degrade topic quality and has a performance as good as BTM.

To verify that our FastBTM converges faster than BTM, we show the log likelihood versus running time in Fig. 6. The second row is the result on the Tweets2011 Collection dataset. We can see that When setting $K = 200$, our algorithm's performance is slightly better than baseline, where the running time of reaching a particular log likelihood is a little less than BTM. When setting $K = 500$, our FastBTM takes $4,900$ seconds to reach a log likelihood of -7.5×10^8, about 1.5 times faster than BTM. The gap between FastBTM and BTM becomes more significant at $K = 1,000$, our FastBTM runs about 5 times faster than BTM on Tweets2011 Collection dataset. A similar result can be

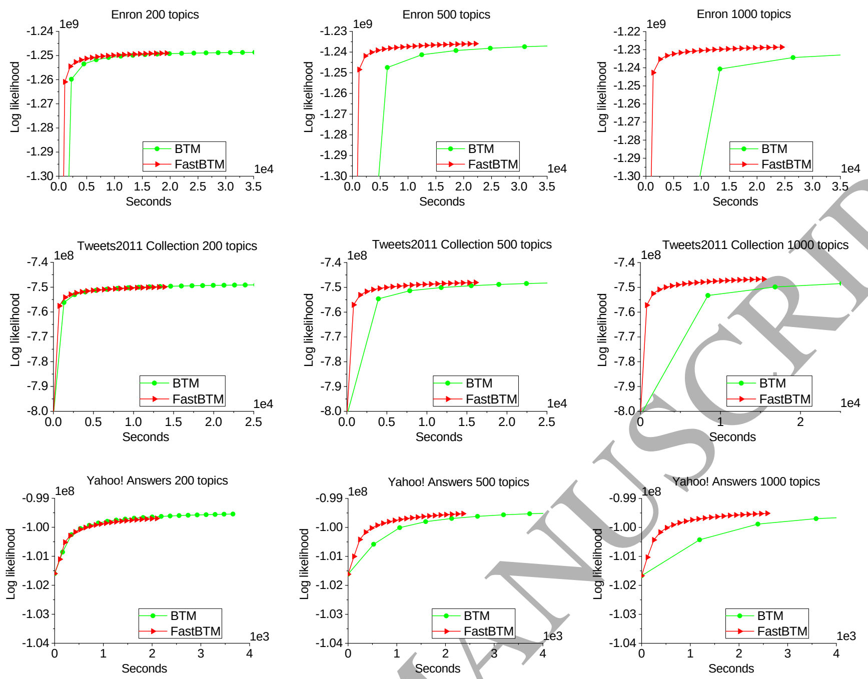


Figure 6: Log likelihood as a function of running time for BTM and FastBTM

observed on Yahoo! Answers dataset. For example, our FastBTM runs a little faster than BTM when setting $K = 200$. When setting $K = 500$ and $K = 1,000$, our FastBTM is about 1 time and 4 times faster than BTM, respectively.

To further illustrate the speedup of our FastBTM, we show the running time for each of the first 200 iterations in Fig. 7. We can see that BTM takes about 70 seconds per iterations, while BTM takes about 130 seconds on Tweets2011 Collection dataset when setting $K = 200$. In other words, FastBTM is about 1 time faster than BTM. With the increase of K, the performance gap between the two algorithms becomes outstanding, where FastBTM is about 3 times and 8 times faster than BTM at $K = 500$ and $K = 1,000$, respectively. Moreover, we can draw the same conclusion from the results on Yahoo! Answers dataset, the third row of Fig. 7. So we can conclude that our FastBTM can reduce the sampling time of short text datasets compared with BTM and when the number

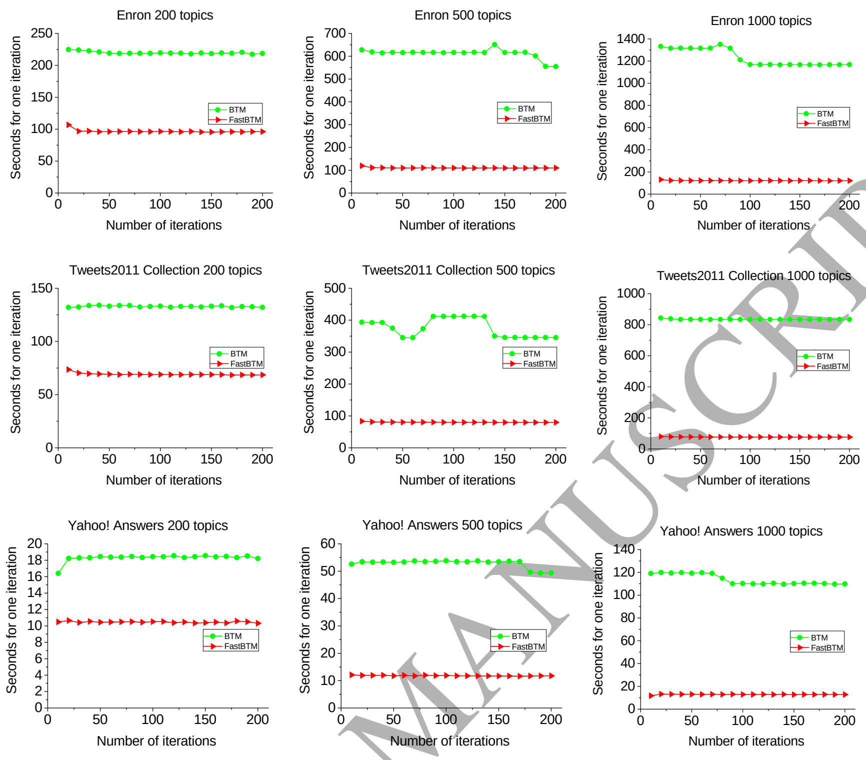


Figure 7: Running time of BTM and FastBTM for each of the first 200 iterations

of topics K increases, the gap in running time speed gets especially larger.

5.4.2. Analysis on long document dataset

In last subsubsection, we have verified the effectiveness of our model on short text datasets. Then we will compare our FastBTM against BTM on the long document dataset. The first row of Fig. 5 shows the convergence results on Enron dataset. We can observe that the performance in convergence of our FastBTM is comparable with BTM demonstrating that our acceleration algorithm doesn't come at the cost of downgrading topic quality on long document dataset.

To establish that our FastBTM converges faster than BTM, we show the log likelihood versus running time on Enron dataset in the first row of Fig. 6. We note that when setting $K = 200$, our algorithm's performance is very close to the baseline, where the running time of reaching a particular log likelihood is a little less than BTM. The performance gap between the two models becomes very prominent at $K = 500$, where our FastBTM takes 3,500 seconds to reach a log likelihood of -1.24×10^9 while BTM takes 16,000 seconds. Namely our algorithm is around 4.5 times as fast as BTM. Unsurprisingly, the performance gap between the two models is extremely significant at $K = 1,000$, where FastBTM is about 6 to 9 times as fast as BTM on Enron dataset.

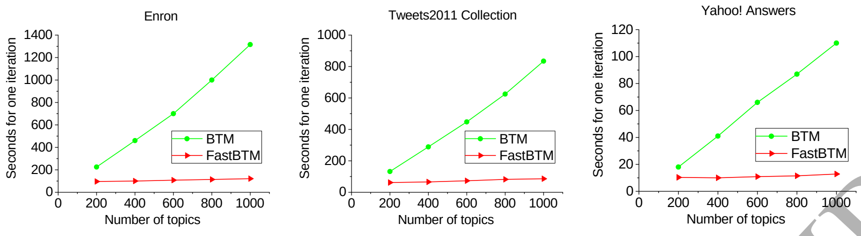
To further demonstrate the performance gap in running time between FastBTM and BTM, we show the running time for each of the first 200 iterations on Enron dataset in the first row of Fig. 7. We observe that our FastBTM consistently outperforms BTM at $K = 200$, $K = 500$ and $K = 1000$. And our FastBTM speed up the sampling process up to 1, 5, 7 times for each topic number. Hence, we conclude that our FastBTM is also effective for long document datasets.

5.4.3. Analysis on the number of topics

From the above, we conclude that with the number of topics K increasing, the gap in running time speed between our FastBTM and BTM gets especially larger. To understand the performance gap completely, we show the running time of one iteration as a function of the number of topics in Fig. 8, where (a), (b) and (c) are the results on Enron dataset, Tweets2011 Collection dataset and Yahoo! Answers dataset, respectively. From (a), we note that the gap in performance scales with increasing number of topics K on long text dataset. Similarly, we can see the same phenomenon on the short text datasets. This confirms the conclusion that our FastBTM takes constant time per sample while BTM takes $O(K)$ operations. What's more, our FastBTM can effectively reduce the sampling time of both short and long text datasets.

5.4.4. Topic Coherence

To demonstrate that our FastBTM doesn't degrade topic quality, we compare the PMI-Score of LDA, BTM and FastBTM on the Tweets2011 dataset with



(a) (b) (c)

Figure 8: Comparison of FastBTM and BTM on Tweets2011 Collection and Enron when varying the number of topics for $K \in 200, 400, 600, 800, 1000$.

Table 2: PMI-Score of different topic model for Tweets2011 dataset. We show the best result in bold font.

K	200			500		
Model	Top5	Top10	Top20	Top5	Top10	Top20
LDA	1.007	1.055	1.014	1.028	1.044	0.913
BTM	1.195	1.139	1.132	**1.187**	1.179	1.123
FastBTM	**1.197**	**1.163**	**1.141**	1.147	**1.185**	**1.175**

the number of most probable words T ranging from 5 to 20 and show the results in Table 2. We can observe that both BTM and FastBTM outperform LDA consistently. At the same time, we find that the PMI-Score of BTM and FastBTM are very close.

To further evaluate the topic quality of our model, we illustrate two topics in Table 3, where the second row is about "iphone" and the third row is about "eat". For each topic, we show the 20 most probable words. What's more, we denote the words irrelevant to the topics in bold font. We can easily observe that both our FastBTM and BTM perform much better than LDA. We can see that BTM and FastBTM achieve almost the same result and the 20 most probable words selected by them are related to "iphone". However, LDA includes some words such as "book", "read" and "write", not relevant with "iphone". For the topic "eat", we can observe the same result. LDA contains some words namely, "day" and "lol" not related to "eat". However, in BTM and FastBTM, only "rt" is not related to "eat".

From Table 2 and 3, we can conclude that the topics discovered by BTM and FastBTM are more coherent than LDA over short texts. What's more, our FastBTM doesn't degrade the topic quality.

Table 3: The 20 most probable words in topics about "iphone" (the second row) and about "eat" (the third row) from Tweets2011 dataset. We show the words irrelevant to the topic in bold font.

LDA	BTM	FastBTM
book ipad iphone app apple free **read** download **reading** ipod android **books** touch store apps **writing write news** blackberry itunes	iphone rt android app ipad apple verizon ipod mobile sony ios touch apps store phone nokia blackberry playstation google samsung	iphone rt android app ipad apple verizon sony phone mobile ipod touch ios blackberry free playstation store apps google nokia
eat pizza chicken food eating chocolate cheese cake breakfast lunch **day** bread taste **lol** cookies rice bacon delicious hot ate	**rt** eat chicken food cheese eating dinner cream lunch breakfast yummy soup chocolate bacon rice salad hot fried cake ice	**rt** eat chicken chocolate cream food ice cheese pizza coffee hot dinner yummy eating cake soup breakfast salad lunch tea

6. Conclusion

In this paper, we propose an acceleration algorithm of BTM, FastBTM, which converges faster than BTM without degrading topic quality. To verify the effectiveness of our FastBTM, we carry out experiments on three datasets, two short text datasets and one long document dataset. Experimentally, we conclude that FastBTM consistently achieves much more significant performance boost relative to the original model. As the number of topics K grows, the gap in running time speed between FastBTM and BTM gets especially larger. And our model successfully reduces sampling complexity of biterm topic model from $O(K)$ to $O(1)$. What's more, our FastBTM is effective for reducing the sampling time of both short text and long text datasets.

Acknowledgment

This work is supported by National Natural Science Foundation of China (Grant No: 61673235).

References

References

[1] W. Wang, H. Xu, X. Huang, Implicit feature detection via a constrained topic model and svm., in: EMNLP, 2013, pp. 903–907.

[2] F. Godin, V. Slavkovikj, W. De Neve, B. Schrauwen, R. Van de Walle, Using topic models for twitter hashtag recommendation, in: Proceedings of the 22nd International Conference on World Wide Web, ACM, 2013, pp. 593–596.

[3] H. Xie, D. Zou, R. Y. Lau, F. L. Wang, T.-L. Wong, Generating incidental word-learning tasks via topic-based and load-based profiles, IEEE multimedia 23 (1) (2016) 60–70.

[4] G. Petrovic, H. Fujita, Semi-automatic detection of sentiment hashtags in social networks, in: International Conference on Intelligent Software Methodologies, Tools, and Techniques, Springer, 2015, pp. 216–224.

[5] K. Zhang, W. Wu, H. Wu, Z. Li, M. Zhou, Question retrieval with high quality answers in community question answering, in: Proceedings of the

23rd ACM International Conference on Conference on Information and Knowledge Management, ACM, 2014, pp. 371–380.

[6] Y. Rao, H. Xie, J. Li, F. Jin, F. L. Wang, Q. Li, Social emotion classification of short text via topic-level maximum entropy model, Information & Management 53 (8) (2016) 978–986.

[7] O. Appel, F. Chiclana, J. Carter, H. Fujita, A hybrid approach to the sentiment analysis problem at the sentence level, Knowledge-Based Systems 108 (2016) 110–124.

[8] T. Hofmann, Probabilistic latent semantic indexing, in: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 1999, pp. 50–57.

[9] D. M. Blei, A. Y. Ng, M. I. Jordan, Latent dirichlet allocation, Journal of machine Learning research 3 (Jan) (2003) 993–1022.

[10] X. Wang, A. McCallum, Topics over time: a non-markov continuous-time model of topical trends, in: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2006, pp. 424–433.

[11] X. Yan, J. Guo, Y. Lan, X. Cheng, A biterm topic model for short texts, in: Proceedings of the 22nd international conference on World Wide Web, ACM, 2013, pp. 1445–1456.

[12] X. Cheng, X. Yan, Y. Lan, J. Guo, Btm: Topic modeling over short texts, IEEE Transactions on Knowledge and Data Engineering 26 (12) (2014) 2928–2941.

[13] P. Wang, H. Zhang, B. Xu, C. Liu, H. Hao, Short text feature enrichment using link analysis on topic-keyword graph, in: Natural Language Processing and Chinese Computing, Springer, 2014, pp. 79–90.

[14] Y. Xia, N. Tang, A. Hussain, E. Cambria, Discriminative bi-term topic model for headline-based social news clustering., in: FLAIRS Conference, 2015, pp. 311–316.

[15] I. Porteous, D. Newman, A. Ihler, A. Asuncion, P. Smyth, M. Welling, Fast collapsed gibbs sampling for latent dirichlet allocation, in: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2008, pp. 569–577.

[16] L. Yao, D. Mimno, A. McCallum, Efficient methods for topic model inference on streaming document collections, in: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2009, pp. 937–946.

[17] A. Q. Li, A. Ahmed, S. Ravi, A. J. Smola, Reducing the sampling complexity of topic models, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2014, pp. 891–900.

[18] J. Yuan, F. Gao, Q. Ho, W. Dai, J. Wei, X. Zheng, E. P. Xing, T.-Y. Liu, W.-Y. Ma, Lightlda: Big topic models on modest computer clusters, in: Proceedings of the 24th International Conference on World Wide Web, ACM, 2015, pp. 1351–1361.

[19] J. Chen, K. Li, J. Zhu, W. Chen, Warplda: a simple and efficient o (1) algorithm for latent dirichlet allocation, in: VLDB, 2016, pp. 744–755.

[20] G. Marsaglia, W. W. Tsang, J. Wang, et al., Fast generation of discrete random variables, Journal of Statistical Software 11 (3) (2004) 1–11.

[21] A. J. Walker, An efficient method for generating discrete random variables with general distributions, ACM Transactions on Mathematical Software (TOMS) 3 (3) (1977) 253–256.

[22] J. Geweke, H. Tanizaki, Bayesian estimation of state-space models using the metropolis–hastings algorithm within gibbs sampling, Computational Statistics & Data Analysis 37 (2) (2001) 151–170.

[23] C. Andrieu, N. De Freitas, A. Doucet, M. I. Jordan, An introduction to mcmc for machine learning, Machine learning 50 (1-2) (2003) 5–43.

[24] W. K. Hastings, Monte carlo sampling methods using markov chains and their applications, Biometrika 57 (1) (1970) 97–109.

[25] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, E. Teller, Equation of state calculations by fast computing machines, The journal of chemical physics 21 (6) (1953) 1087–1092.

[26] L. Tierney, Markov chains for exploring posterior distributions, the Annals of Statistics (1994) 1701–1728.

[27] T. Griffiths, Gibbs sampling in the generative model of latent dirichlet allocation, Tech. rep., Stanford University (2002).

[28] T. L. Griffiths, M. Steyvers, Finding scientific topics, Proceedings of the National academy of Sciences 101 (suppl 1) (2004) 5228–5235.

[29] J. K. Pritchard, M. Stephens, P. Donnelly, Inference of population structure using multilocus genotype data, Genetics 155 (2) (2000) 945–959.

[30] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, R. Harshman, Indexing by latent semantic analysis, Journal of the American society for information science 41 (6) (1990) 391.

[31] D. Ramage, D. Hall, R. Nallapati, C. D. Manning, Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora, in: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1, Association for Computational Linguistics, 2009, pp. 248–256.

[32] D. Ramage, S. T. Dumais, D. J. Liebling, Characterizing microblogs with topic models., ICWSM 10 (2010) 1–1.

[33] D. Ramage, C. D. Manning, S. Dumais, Partially labeled topic models for interpretable text mining, in: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2011, pp. 457–465.

[34] M. Rosen-Zvi, T. Griffiths, M. Steyvers, P. Smyth, The author-topic model for authors and documents, in: Proceedings of the 20th conference on Uncertainty in artificial intelligence, AUAI Press, 2004, pp. 487–494.

[35] W. X. Zhao, J. Jiang, J. Weng, J. He, E.-P. Lim, H. Yan, X. Li, Comparing twitter and traditional media using topic models, in: European Conference on Information Retrieval, Springer, 2011, pp. 338–349.

[36] J. Xu, P. Liu, G. Wu, Z. Sun, B. Xu, H. Hao, A fast matching method based on semantic similarity for short texts, in: Natural Language Processing and Chinese Computing, Springer, 2013, pp. 299–309.

[37] V. W. Chu, R. K. Wong, C. H. Chi, P. C. Hung, Web service orchestration topic mining, in: Web Services (ICWS), 2014 IEEE International Conference on, IEEE, 2014, pp. 225–232.

[38] Y. Pan, J. Yin, S. Liu, J. Li, A biterm-based dirichlet process topic model for short texts, in: Proceedings of 3rd International Conference on Computer Science and Service System (CSSS 2014), 2014, pp. 301–304.

[39] X. Yan, J. Guo, Y. Lan, J. Xu, X. Cheng, A probabilistic model for bursty topic discovery in microblogs., in: AAAI, 2015, pp. 353–359.

[40] W. Chen, J. Wang, Y. Zhang, H. Yan, X. Li, User based aggregation for biterm topic model, Volume 2: Short Papers (2015) 489.

[41] J. Li, H. Xu, User-ibtm: An online framework for hashtag suggestion in twitter, in: International Conference on Web-Age Information Management, Springer, 2016, pp. 279–290.

[42] G. Heinrich, Parameter estimation for text analysis, University of Leipzig, Tech. Rep.

[43] M. D. Vose, A linear algorithm for generating random numbers with a given distribution, IEEE Transactions on software engineering 17 (9) (1991) 972–975.

[44] D. Newman, J. H. Lau, K. Grieser, T. Baldwin, Automatic evaluation of topic coherence, in: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Association for Computational Linguistics, 2010, pp. 100–108.

[45] D. Newman, S. Karimi, L. Cavedon, External evaluation of topic models, in: in Australasian Doc. Comp. Symp., 2009, Citeseer, 2009.