Topic Modeling for Short Texts with Auxiliary Word Embeddings

Chenliang Li[1], Haoran Wang[1], Zhiqian Zhang[1], Aixin Sun[2], Zongyang Ma[2]
[1]State Key Lab of Software Engineering, Computer School, Wuhan University, China
cllee@whu.edu.cn,whrwhu@gmail.com,zhangzq2011@whu.edu.cn
[2]School of Computer Science and Engineering, Nanyang Technological University, Singapore
{axsun,zyma}@ntu.edu.sg

ABSTRACT

For many applications that require semantic understanding of short texts, inferring discriminative and coherent latent topics from short texts is a critical and fundamental task. Conventional topic models largely rely on word co-occurrences to derive topics from a collection of documents. However, due to the length of each document, short texts are much more sparse in terms of word co-occurrences. Data sparsity therefore becomes a bottleneck for conventional topic models to achieve good results on short texts. On the other hand, when a human being interprets a piece of short text, the understanding is not solely based on its content words, but also her background knowledge (*e.g.,* semantically related words). The recent advances in word embedding offer effective learning of word semantic relations from a large corpus. Exploiting such auxiliary word embeddings to enrich topic modeling for short texts is the main focus of this paper. To this end, we propose a *simple*, *fast*, and *effective* topic model for short texts, named GPU-DMM. Based on the Dirichlet Multinomial Mixture (DMM) model, GPU-DMM promotes the semantically related words under the same topic during the sampling process by using the generalized Pólya urn (GPU) model. In this sense, the background knowledge about word semantic relatedness learned from millions of external documents can be easily exploited to improve topic modeling for short texts. Through extensive experiments on two real-world short text collections in two languages, we show that GPU-DMM achieves comparable or better topic representations than state-of-the-art models, measured by topic coherence. The learned topic representation leads to the best accuracy in text classification task, which is used as an indirect evaluation.

Categories and Subject Descriptors

I.2.7 [**Natural Language Processing**]: Text analysis

Keywords

Topic Model, Short Texts, Word Embeddings

SIGIR '16, July 17-21, 2016, Pisa, Italy

DOI: http://dx.doi.org/10.1145/2911451.2911499

1. INTRODUCTION

Short texts have become a fashionable form of information on the Internet. Examples include web page snippets, news headlines, text advertisements, tweets, status updates, and questions/answers, to name a few. Given the large volume of short texts available, effective and efficient models to discover the latent topics from short texts become fundamental to many applications that require semantic understanding of textual content, such as user interest profiling [38], topic detection [37], comment summarization [31], content characterizing [32], and classification [34].

Conventional topic modeling techniques, *e.g.,* pLSA and LDA, are widely used to infer latent topical structure from text corpus [2, 12]. In these models, each document is represented as a multinomial distribution over topics and each topic is represented as a multinomial distribution over words. Statistical techniques (*e.g.,* Gibbs sampling) are then employed to identify the underlying topic distribution of each document as well as word distribution of each topic, based on the high-order word co-occurrence patterns [29]. These models and their variants have been studied extensively for various tasks in information retrieval and text mining [14, 36, 42]. Despite their great success on many tasks, conventional topic models experience a large performance degradation over short texts because of limited word co-occurrence information in short texts. In other words, data sparsity impedes the generation of discriminative document-topic distributions, and the resultant topics are less semantically coherent.

Several ingenious strategies have been proposed to deal with the data sparsity problem in short texts. One strategy is to aggregate a subset of short texts to form a longer pesudo-document. Conventional topic models are then applied over these pesudo-documents. The aggregation is often guided by auxiliary metadata information. For example, in the context of Twitter, tweets can be aggregated based on their hashtags, users, locations, or timestamps before applying LDA [13, 19, 38]. However, a limitation of this strategy is that additional metadata may not be available always, *e.g.,* web page snippets. Another strategy is to restrict the document-topic distribution, such that each short text is sampled from a single topic, known as mixture of unigrams or Dirichlet Multinomial Mixture (DMM) model [26,40,42]. Given the limited content in a short text, this simplification is reasonable and it alleviates the data sparsity problem to some extent. It is reported to be a better alternative to conventional LDA models [39,42]. The third strategy is to design a brand new topic model by explicitly incorporating additional word co-occurrence information. Examples include modeling word co-occurrence patterns [39] and using soft-clustering mechanism for further word co-occurrence augmentation [30]. However, the word co-occurrence information that can be captured by these models

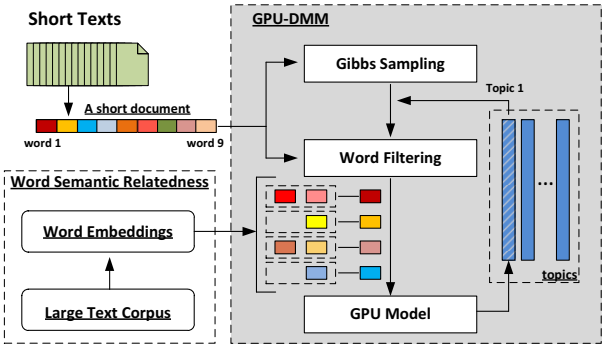


Figure 1: GPU-DMM Overview

is limited to the short text corpus itself. Given two words having strong semantically relatedness but rarely co-occurring in a short text corpus, these models can not fully capture the semantic relatedness between the two words.

When a human being interprets a piece of text, the understanding is not solely based on its content, but also her background knowledge, *e.g.,* semantic relatedness between words. It is also natural to exploit external lexical knowledge to guide the topic inference over short texts. Existing works in this line largely rely on either external thesauri (*e.g.,* WordNet) or lexical knowledge derived from documents in a specific domain (*e.g.,* product comments) [5–7]. The availability of such knowledge becomes vital for these models. This calls for a more generic model that can be effectively applied to short texts, without the need of manually constructed thesauri, and not limited to external documents in specific domains.

In this paper, we propose a new topic model for short texts, named GPU-DMM. GPU-DMM is designed to leverage the general word semantic relatedness knowledge during the topic inference process, to tackle the data sparsity issue. Effective learning of general word semantic relatedness is now feasible and practical with the recent development in neural network techniques, which have contributed improvements in many tasks in Information Retrieval (IR) and Natural Language Processing (NLP) [9, 10, 16, 17, 43]. Specifically, neural network language models, *e.g.,* Continuous Bag-of-Words (CBOW), Continuous Skip-gram model, and Glove model [20, 27], learn word vectors (also called *word embeddings*) with the aim of fully retaining the contextual information for each word, including both semantical and syntactical relations. The shallow neural network structure designed in these techniques is computationally effective on large text corpus. For instance, training skip-gram based word embeddings on a Google News corpus with 100 billion words takes less than one day on a modest computer.[1] That is, general word semantic relatedness knowledge can be efficiently learned from a very large text corpus, in any language. In fact, there are many pre-trained word embeddings learned from resources like Wikipedia, Twitter, and Freebase, publication available on the Web.[2]

Shown in Figure 1, the proposed GPU-DMM extends the Dirichlet Multinomial Mixture (DMM) model by incorporating the learned word relatedness from large text corpus through the *generalized Pólya urn* (GPU) model [18] in topic inferences. More specifically, GPU-DMM promotes the semantically relevant words under the same topic after sampling a topic for a short document. In this sense, GPU-DMM links the semantically relevant words together, even if they share very limited or no co-occurrences in the current collection of short texts being modeled. A filtering strategy is also introduced in GPU-DMM to guide the topic inference process, such that only appropriate external knowledge is exploited for the sampled topic. Because GPU-DMM takes in the word embeddings that have been learned from external documents, the model is *fast* and *flexible* in taking in word embeddings learned from any other large text collections. On two real-world datasets in two different languages (*i.e.,* snippets of an English search engine, and questions from Q&A service in Chinese), GPU-DMM discovers more prominent topics and achieves better classification accuracy than existing state-of-the-art alternatives. The main contributions of this paper are summarized as follows:

1. We develop a simple, fast, and effective topic model to learn the latent topic patterns over short texts. The model is flexible in directly taking in word semantic relations learned from large text corpus, which is domain free and ease to access. To the best of our knowledge, this is the first work for a topic model to incorporate the general word semantic relatedness knowledge based on word embeddings with GPU model.

2. On two real-world short text collections in two languages, we evaluate the proposed GPU-DMM against a few state-of-the-art alternatives for short texts. Experimental results demonstrate our model's superiority, in topic coherence, text classification accuracy, and learning speed.

3. We empirically study the impact of two document representation inference methods. Our results suggest that the summation over each word's contribution within a short document is more appropriate for topic-focused downstream applications, *e.g.,* text classification.

2. RELATED WORK

We review recent advances on learning better topic representations on short texts. We then focus on the models with word embeddings because our model uses word embeddings as external knowledge.

Topic Models for Short Texts. Conventional topic models such as pLSA and LDA are designed to implicitly capture word co-occurrence patterns at document-level, to reveal topic structures. Thus more word co-occurrences would lead to more reliable and better topic inference. Because of the length of each document, conventional topic models suffer a lot from the data sparsity problem in short texts, leading to inferior topic inferences. Earlier studies focus on exploiting external knowledge to help refine the topic inference of short texts. Phan *et al.* proposed to infer topic structure of short texts by using the learnt latent topics from Wikipedia [28]. Similarly, Jin *et al.* infer latent topics of short texts for clustering by using auxiliary long texts [15]. These models require a large regular text corpus of high quality, which may not be always available in some domains and/or languages.

Given the limited context information in short texts, many aggregation strategies have been studied by merging short texts into long pseudo-documents. Conventional topic modeling is then applied to infer the latent topics. In [38], the authors aggregate tweets from the same user as a pseudo-document before performing standard LDA model. Other metadata that have been used for short text aggregation include hashtags, timestamps, and named entities [13, 19, 42]. However, favorable metadata may not be available in some domains, *e.g.,* search snippets and news headlines. These studies suggest that topic models specifically designed for general short texts are imperative.

[1] https://code.google.com/p/word2vec

[2] Details are listed at https://github.com/3Top/word2vec-api

A simple and effective topic model, named Dirichlet Multinomial Mixture (DMM) model, has been employed to discover latent topics in short texts in many tasks [40, 42]. DMM is based on the assumption made in the mixture of unigrams model proposed by Nigam *et al.* [26], *i.e.,* each document is sampled from a single latent topic. Given the limited content of short texts, this assumption is reasonable and is proven to be more effective than conventional topic models in many studies [30, 39, 42]. Yin and Wang [40] propose a collapsed Gibbs Sampling algorithm for DMM and show its effectiveness in short text clustering. Due to its simplicity and effectiveness, we develop GPU-DMM on the basis of DMM, as its name suggests.

Recently, many efforts have been spent towards intensifying the word co-occurrence information from the collection of short texts being modeled. Yan *et al.* propose a novel biterm topic model (BTM) to explicitly model the generation of word co-occurrence patterns instead of single words as do in many topic models [39]. Their experimental results show that BTM produces discriminative topic representations as well as more coherent topics for short texts. Inspired by the aforementioned aggregation strategies, Quan *et al.* propose a self-aggregation based topic model (SATM) for short texts [30]. SATM assumes that each short text is a segment of a long pseudo-document and shares the same topic proportion of the latter. The topic inference and the aggregation process is conducted in a mutual reinforcement manner, such that the aggregation is based on the topical similarity of the texts and vice versa. However, setting an appropriate number of long pseudo-documents in SATM is not an easy task. Further, the inference process involving both text aggregation and topic sampling is time-consuming.

Topic Models for Short Texts with Word Embeddings. Word embeddings, first introduced in [33], have been successfully applied in language models and many NLP tasks, such as named entity recognition and parsing [1, 22]. Word embeddings are useful because they encode both syntactic and semantic information of words into continuous vectors and similar words are close in vector space.

Most relevant to ours is the work by Nguyen *et al.* [24]. They propose a topic model with word embeddings for short texts, called LF-DMM. Built based on DMM, LF-DMM replaces the topic-word multinomial distribution with a two-component mixture of a Dirichlet multinomial component and a continuous word embedding component. That is, each word in a short text is generated from either the Dirichlet multinomial distribution or the probability estimated by using word embeddings with respect to the sampled topic. A switch variable is utilized to decide which component is used to generate a word. Rather than using Bayesian parameter estimation, a simple and hard coded switch probability value is used for the switch variable in LF-DMM. This simple switch mechanism could incur some noise into the inference process. In order to estimate the word embedding component of each word, the topics are projected into the same latent continuous space as word embeddings by optimizing a regularized log-linear model. However, this optimization process is computational expensive. Similarly, Das *et al.* propose a LDA based topic model by using multivariate Gaussian distributions with word embeddings [11]. Our work differs significantly from these studies. GPU-DMM exploits the general word semantic relatedness knowledge provided by auxiliary word embeddings by using the generalized Pólya urn model in the topic inference of short texts. Compared with existing approaches of incorporating word embeddings in topic model, GPU reduces the computational cost significantly. To the best of our knowledge, GPU-DMM is the first attempt to combine word embeddings and GPU model for solving sparsity problem of short texts.

3. GPU-DMM

As its name suggests, the proposed GPU-DMM model is built upon the Dirichlet Multinomial Mixture (DMM). Given a short document, GPU-DMM samples a topic for it based on the conditional probabilities similar to DMM. The words that are highly relevant to the topic are then selected, and their semantically related words are extracted and promoted by using the GPU model [18]. As shown in Figure 1, the auxiliary word embeddings utilized in GPU-DMM is pre-learned using the state-of-the-art word embedding techniques from large document collections. Next, we present the details of the proposed model GPU-DMM.

3.1 Dirichlet Mixture Model

Dirichlet Mixture Model is a generative probabilistic model with the assumption that *a document is generated from a single topic* [26, 40]. That is, all the words within a document are generated by using the same topic distribution.

Given a short text corpus of D documents, with a vocabulary of size V, and K pre-defined latent topics, each document d is associated with a specific topic k. Then the N_d words in document d $\{w_{d,1}, w_{d,2}, \ldots, w_{d,N_d}\}$ are generated by the topic-word multinomial distribution $p(w|z = k) = \phi_k$ by assuming independence of the words themselves. More formally, with the Dirichlet priors of α and β, the generative process of DMM is described as follows:

1. Sample a topic proportion $\theta \sim Dirichlet(\alpha)$
2. For each topic $k \in \{1, ..., K\}$
   Draw a topic-word distribution $\phi_k \sim Dirichlet(\beta)$
3. For each document $d \in \{1, ..., D\}$
   (a) Sample a topic $z_d \sim Multinomial(\theta)$
   (b) For each word $w \in \{w_{d,1}, ..., w_{d,N_d}\}$
   Sample a word $w \sim Multinomial(\phi_{z_d})$

The hidden variables in the generative process can be approximated by applying Gibbs sampling. Following the approach in [40], a topic z is sampled for each document in every iteration according to the following conditional distribution:

$$p(z_d = k|\vec{z}_{\neg d}, \vec{d}) \propto \frac{m_{k,\neg d} + \alpha}{D - 1 + K\alpha} \times \frac{\prod_{w \in d} \prod_{j=1}^{N_d^w} (n_{k,\neg d}^w + \beta + j - 1)}{\prod_{i=1}^{N_d} (n_{k,\neg d} + V\beta + i - 1)} \quad (1)$$

where N_d^w is term frequency of word w in document d, $m_{k,\neg d}$ is the number of documents assigned to topic k, $n_{k,\neg d}^w$ is the number of times that word w is assigned to topic k, and $n_{k,\neg d} = \sum_w^V n_{k,\neg d}^w$. Symbol $\neg d$ means that document d is excluded from the counting. The posterior distribution is calculated by using point estimation:

$$p(w|z = k) = \frac{n_k^w + \beta}{\sum_w^V n_k^w + V\beta} \quad (2)$$

3.2 Auxiliary Word Embeddings

Because of the length of short texts, the words with high semantic relatedness may not frequently co-occur in the same short texts. On the other hand, because the word embeddings are learnt with the aim to retain words' global contextual information, the learnt word embeddings capture the general word co-occurrence patterns. That is, words that are semantically or syntactically similar to

Table 1: Example words and their co-occurrences with semantically related words in the Snippet dataset.

Example (Freq.)	Word (Freq., #co-occur with example word)	
artworks (20)	paintings (17, 1)	artist (53, 3)
movie (333)	cinema (33, 10)	filmography (19, 7)
fiction (45)	book (142, 3)	literary (12, 2)
company (143)	business (431, 16)	industry (117, 4)
stock (131)	price (39, 1)	equity (15, 2)

each other are projected to be closer in the latent space. Next, we illustrate this point using some example words.

Table 1 lists 5 example words in the left-hand column (*i.e.,* "artworks", "movie", "fiction", "company", "stock"). The number following each word in paraphrase is the document frequency of the word in the Web snippet dataset. This short text dataset contains $12,340$ Web search snippets (see Section 4.1 for more details). For example, "artworks (20)" means that the word "artworks" appears in 20 short documents in the Snippet dataset. In the right-hand column, for each word, we show two semantically related words obtained from pre-trained word embeddings. The word embeddings used here are learned from the aforementioned Google News corpus with 100 billion words of vocabulary size of 3 million. The two numbers following each word are (i) the document frequency of the word, and (ii) the number of co-occurrences with the example word on the left-hand side, in the Web snippet dataset. For instance, "movie" appears in 333 snippets, "cinema" appears in 33 snippets, and the two word co-occur in 10 snippets.

Shown in the table, the words obtained from word embeddings indeed show strong semantic relatedness with the example words. More importantly, the semantic relations provided by word embeddings are much broader than the limited lexical relations defined in an external thesauri (*e.g.,* synonymy, antonym and adjective-attribute relations in WordNet). The table also shows that semantically related words may not co-occur frequently in the short text collection. We therefore believe that incorporating auxiliary word embeddings learned from large corpus would significantly enhance topic modeling on short texts.

3.3 Incorporating Word Embeddings by GPU

It has been validated that the topic coherence measure based on word co-occurrence pattern is a reliable indicator of topic quality and is highly consistent with the human expert annotation [21]. Accordingly, it is reasonable that the words with high semantic relatedness should be clustered together under the same topic. In the following, we present how GPU-DMM achieves this through the generalized Pólya urn model [18].

Generalized Pólya Urn Model can be understood in terms of colored balls in a urn, where the probability of seeing a ball of each color is linearly proportional to the number of balls of that color in the urn. In a simple Pólya urn model, when a ball of a particular color is sampled from the urn, the sampled ball along with a new ball of that color is put back into the urn. This process is equivalent to the Gibbs sampling process utilized for the topic inference with Dirichlet-Multinomial distribution under the i.i.d assumption. In the generalized Pólya urn model, when a ball of a particular color is sampled, a certain number of balls of similar colors are put back along with the original ball and a new ball of that color. In this sense, the set of balls of similar colors are promoted as a whole from this iterative process. By analogy with the GPU model, in our case, given a ball of word w, the balls of similar colors refer to the semantically related words to word w. As the result, sampling a word w in topic t not only increases the probability of w itself under topic t, but also increases the association between the topic and w's semantically related words. Several existing works exploit the GPU model and the external thesauri or domain-specific knowledge for better topic inference of the standard LDA [5–7, 21]. However, these works rely on the specific domain knowledge, which may restrict their usage in a broader range. Here, we exploit the combination of GPU model and the global word relatedness knowledge, leading to a generic solution for short text topic modeling.

Formally, given pre-trained word embeddings, we measure the semantic relatedness between two words w and w' by the cosine similarity between their vector representations in the latent space (*i.e.,* the word embeddings of the two words). The semantic relatedness between the word pair is denoted by $sr(w, w')$. Then a word semantic relatedness matrix $\mathbb{M}$ can be constructed, consisting of all word pairs whose semantic relatedness score is higher than a predefined threshold ϵ, *i.e.,* $\mathbb{M} = \{\langle w_i, w_j\rangle | sr(w_i, w_j) > \epsilon\}$. The threshold ϵ is set to filter less semantic related word pairs.

Being a preliminary study on exploiting general word semantic relatedness knowledge based on word embeddings, we fix the amount of promotion μ for each semantically related word w' when working on word w. The promotion matrix $\mathbb{A}$ with respect to each word pair is defined below, where $\mathbb{M}_w$ is the row in $\mathbb{M}$ corresponding to word w. Note that $\mathbb{M}_w$ includes the word w itself.

$$\mathbb{A}_{w,w'} = \begin{cases} 1 & w{=}w' \\ \mu & w' \in \mathbb{M}_w \text{ and } w' \neq w \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The weight μ could be set based on the auxiliary word embeddings under use.

Incorporating the GPU model with DMM gives us the proposed GPU-DMM model. However, simply taking all word pairs in matrix $\mathbb{M}$ may not be the best choice for topic modeling in short texts, explained next.

Word Filtering. Based on DMM, GPU-DMM samples a short text from a single topic. All words in the short text are assigned to the same topic. Take a short text *"info website web cern consortium server project world copy first"* as an example. This document can be assigned to a computer related topic. However, words like *consortium* and *first*, along with their semantically related words, could be irrelevant to computer related topics. This scenario is prevalent in many short texts and could be harmful to DMM based topic models. In this sense, simply adopting GPU model for every word in a document to promote their semantically related words under the sampled topic could adversely affect the quality of the topic. This calls for an appropriate strategy to reinforce only the semantically related words if and only if a word has strong ties with the sampled topic. To this end, we propose a nonparametric probabilistic sampling strategy as follows:

$$\mathbb{S}_{d,w} \sim Bernoulli(\lambda_{w,z_d}) \tag{4}$$

$$\lambda_{w,z} = \frac{p(z|w)}{p_{max}(z'|w)} \tag{5}$$

$$p_{max}(z|w) = \max_k p(z = k|w)$$

$$p(z = k|w) = \frac{p(z = k)p(w|z = k)}{\sum_{i=1}^{K} p(z = i)p(w|z = i)} \tag{6}$$

In Equation 4, $\mathbb{S}_{d,w}$ indicates whether GPU is applied to word w given document d and topic z_d, where λ_{w,z_d} is defined in Equation 5. Observe from Equation 5, $\mathbb{S}_{d,w}$ is strongly related to the ratio of the conditional topic probability z_d given word w to its

Algorithm 1: GPU-DMM

input : Topic number K, α, β, μ, $\mathbb{M}$ and D short documents
output: The posterior topic-word distribution

1 **foreach** $d \in D$ **do**
2 $\quad z_d \leftarrow z \sim Multinomial(1/K)$;
3 $\quad m_z \leftarrow m_z + 1$;
4 $\quad \tilde{n}_z \leftarrow \tilde{n}_z + 1$;
5 $\quad$ **foreach** $w \in d$ **do**
6 $\quad\quad \tilde{n}_z^w \leftarrow \tilde{n}_z^w + N_d^w$;
7 $\quad\quad \mathbb{S}_{d,w} \leftarrow 0$;
8 **foreach** *iteration* **do**
9 $\quad$ UpdateWordTopicProb() ; /* See Eq. 6 and 8 */
10 $\quad$ **foreach** $d \in D$ **do**
11 $\quad\quad z \leftarrow z_d$;
12 $\quad\quad m_z \leftarrow m_z - 1$;
13 $\quad\quad$ **foreach** $w \in d$ **do**
14 $\quad\quad\quad$ UpdateCounter ($\mathbb{S}_{d,w}$, $\mathbb{A}$, d, w, $False$);
15 $\quad\quad z_d \leftarrow z \sim p(z_d = z|\vec{z}_{\neg d}, \vec{d})$;
16 $\quad\quad m_z \leftarrow m_z + 1$;
17 $\quad\quad$ **foreach** $w \in d$ **do**
18 $\quad\quad\quad$ UpdateGPUFlag ($\mathbb{S}_{d,w}$); /* See Eq. 4 */
19 $\quad\quad\quad$ UpdateCounter ($\mathbb{S}_{d,w}$, $\mathbb{A}$, d, w, $True$);

Algorithm 2: UpdateCounter ($\mathbb{S}_{d,w}$, $\mathbb{A}$, d, w, $promotion$)

1 **if** $promotion == True$ **then**
2 $\quad$ **if** $\mathbb{S}_{d,w} == 1$ **then** /* To apply GPU */
3 $\quad\quad$ **foreach** $w' \in \mathbb{M}_w$ **do**
4 $\quad\quad\quad \tilde{n}_z \leftarrow \tilde{n}_z + N_d^w \cdot \mathbb{A}_{w,w'}$;
5 $\quad\quad\quad \tilde{n}_z^{w'} \leftarrow \tilde{n}_z^{w'} + N_d^w \cdot \mathbb{A}_{w,w'}$;
6 $\quad$ **else**
7 $\quad\quad \tilde{n}_z \leftarrow \tilde{n}_z + 1$;
8 $\quad\quad \tilde{n}_z^w \leftarrow \tilde{n}_z^w + 1$;
9 **else**
10 $\quad$ **if** $\mathbb{S}_{d,w} == 1$ **then** /* GPU was applied */
11 $\quad\quad$ **foreach** $w' \in \mathbb{M}_w$ **do**
12 $\quad\quad\quad \tilde{n}_z \leftarrow \tilde{n}_z - N_d^w \cdot \mathbb{A}_{w,w'}$;
13 $\quad\quad\quad \tilde{n}_z^{w'} \leftarrow \tilde{n}_z^{w'} - N_d^w \cdot \mathbb{A}_{w,w'}$;
14 $\quad$ **else**
15 $\quad\quad \tilde{n}_z \leftarrow \tilde{n}_z - 1$;
16 $\quad\quad \tilde{n}_z^w \leftarrow \tilde{n}_z^w - 1$;

maximal topic probability. That is, if word w is highly relevant to topic z in terms of $p(z|w)$, GPU model is more likely to be applied to w when the sampled topic of its document is indeed z.

Because Gibbs sampling is a stochastic process (*i.e.,* the topic of a document is randomly sampled), there is a chance that an incorrect topic is sampled for a document. However, given the dominant topic of a document, the semantically related words of an irrelevant word are less likely to be promoted under the topic, because of the above probabilistic sampling strategy. The adverse impact of an incorrect sampled topic can not be aggravated by using GPU model. In the proposed word filtering strategy, calculating $p(z = k|w)$ (see Equation 6) for every word w of a document and topic k is very time-consuming. For efficiency purpose, we calculate $p(z = k|w)$ at the beginning of each iteration, and use these static values during the whole iteration, detailed next.

Model Inference. GPU-DMM and DMM model share the same generative process and graphical representation, but differ in topic inference process. In GPU-DMM, the *GPU-based Gibbs sampling* is applied during the topic inference process.

The GPU model is nonexchangeable, which suggests that the joint probability of the words under a specific topic is not invariant to the permutation of those words. This results in a more complex inference process. Following the work of Mimno *et al.* [21], we approximate the true Gibbs sampling distribution by treating each word as if it was the last word, ignoring its implications for subsequent words and their topic assignments. Accordingly, the conditional distribution for Gibbs sampling in Equation 1 is rewritten for GPU-DMM as follows:

$$p(z_d = k|\vec{z}_{\neg d}, \vec{d}) \propto \frac{m_{k,\neg d} + \alpha}{D - 1 + K\alpha} \times \frac{\prod_{w \in d} \prod_{j=1}^{N_d^w} (\tilde{n}_{k,\neg d}^w + \beta + j - 1)}{\prod_{i=1}^{N_d} (\tilde{n}_{k,\neg d} + V\beta + i - 1)} \quad (7)$$

In the Equation 7, $\tilde{n}_{k,\neg d}$ is the number of words associated with topic k, $\tilde{n}_{k,\neg d}^w$ is the number of balls of word w in the urn of topic k, $m_{k,\neg d}$ is the number of documents associated with topic k, symbol $\neg d$ means that document d is excluded from the counting, same as in Equation 1. The posterior distribution in Equation 2 for GPU-DMM is rewritten in Equation 8:

$$p(w|z = k) = \frac{\tilde{n}_k^w + \beta}{\sum_w^V \tilde{n}_k^w + V\beta} \quad (8)$$

The details of the Gibbs sampling process of GPU-DMM is described in Algorithm 1. At first, GPU-DMM initializes the topic assignment for each document with a uniform distribution without applying the GPU model (Lines 1-6). This initialization process is the same as in DMM. In each iteration of Gibbs sampling, we firstly calculate the conditional topic distribution, $p(z = k|w)$ for each k and w, based on Equations 6 and 8 (Line 9). These values will be used during the whole iteration. Then, the topic of each document d is resampled based on the conditional distribution in Equation 7 (Lines 10-14). During this process, GPU-DMM subtracts the corresponding counts for each word w based on the flag $\mathbb{S}_{d,w}$ in the previous iteration by calling the function *UpdateCounter()* (Line 14, Equation 4).

Shown in Algorithm 2, if $\mathbb{S}_{d,w} = 1$, meaning that the related words $\mathbb{M}_w$ for word w have been promoted in the last iteration, then the corresponding counts of each related word $w' \in \mathbb{M}_w$ are subtracted proportional to $\mathbb{A}_{w,w'}$ (Lines 11-13 in Algorithm 2). Recall that the semantic relatedness knowledge $\mathbb{M}_w$ of word w also contains w itself, *i.e.,* $w \in \mathbb{M}_w$ (see Equation 3). The substraction process is applied to both w itself and its semantically related words. Otherwise, if the related words were not promoted in the last iteration, then simple subtraction is applied to w as the same in DMM (Lines 15, 16 in Algorithm 2).

With the new sampled topic, the flag $\mathbb{S}_{d,w}$ for each word w is updated based on Equation 4 (Line 18 in Algorithm 1). Then the corresponding counts for word w are added based on the updated flag $\mathbb{S}_{d,w}$ through function *UpdateCounter()* (Line 19 in Algorithm 1). Similarly, if $\mathbb{S}_{d,w} = 1$, w and all its related words are promoted. Otherwise, simple update as in DMM is applied (Lines 2-8 in Algorithm 2)

This iterative process continues until the predefined number of iterations is reached.

Model Complexity. We now analyze the time complexity of GPU-DMM with reference to DMM. The time complexity of DMM in an iteration is $O(KD\bar{\ell})$, where K is the number of topics, D is the number of documents, and $\bar{\ell}$ is the average document length. Being extended with GPU model, GPU-DMM has a time complexity of $O(KD\bar{\ell}+D\bar{\ell}\tau+KV)$, where τ is a coefficient by considering the cost involved by applying the GPU model. The τ value depends on: (i) the average number of words for which GPU model is applied based on Equation 4, and (ii) the average number of semantically related words for each word appear in the short text corpus. KV is the computation required for calculating all $p(z = k|w)$.

Note that the size of the semantic relatedness knowledge $\mathbb{M}_w$ for word w obtained from auxiliary word embeddings could be large, but not all these semantically related words appear in the underlying short texts being modeled. In this sense, τ is expected to be relatively small. It is expected that $KV < KD\bar{\ell}$. Hence, GPU-DMM does not add too much computational cost to DMM.

4. EXPERIMENT

In this section, we conduct extensive experiments to evaluate the proposed GPU-DMM against the state-of-the-art alternatives.[3] The performance in terms of topic coherence and document classification are reported over two publicly available datasets, *i.e.,* an English Web search snippet dataset and a Chinese Q&A dataset. We also report the time taken per iteration for all models evaluated in our experiments. The experimental results show that our proposed model provides promising performance in both effectiveness and efficiency.

4.1 Datasets

BaiduQA dataset is a collection of $648,514$ questions crawled from a popular Chinese Q&A website[4]. Each question is annotated with a category label by its asker. This dataset was prepared and has been previously used in [8, 39]. The dataset was preprocessed by the authors, *e.g.,* Chinese word segmentation and duplicate words removal. That is, every word has frequency of 1 in the question containing it. In our experiments, we removed the extremely short questions that contain only a single word.

Web Snippet (**Snippet** for short) dataset contains $12,340$ Web search snippets. This dataset has been used in a few studies [4, 28,35]. Each snippet belongs to one of 8 categories. We performed the following preprocessing on this dataset: (1) convert letters to lowercase; (2) remove all non-alphabetic characters and stop words[5]; (3) remove the words with fewer than 3 characters; (4) remove words with document frequency less than 3 in the dataset. As in the BaiduQA dataset, we further remove duplicate words within each snippet, reinforcing its data sparsity.

Statistics on the two datasets after preprocessing is reported in Table 2. Observe that the BaiduQA dataset is much more sparser with a larger vocabulary and much shorter documents compared to the Snippet dataset.

4.2 Experimental Setup

Word Embeddings. For the Snippet dataset, we use the pre-trained 300-dimensional word embeddings from the Google News corpus.[6] For the BaiduQA dataset, we train 100-dimensional word embeddings from 7 million Chinese articles crawled from Baike website,[7] using Goolge's Word2Vec toolkit with Skip-gram algorithm [20]. If a word has no embedding, the word is considered as having no word semantic relatedness knowledge.

Table 2: Statistics on the two datasets. #Label: the number of ground truth labels or categories; #Docs: the total number of documents; #Words: average number of words per document.

Dataset	#Label	#Docs	#Words	Vocabulary
Snippet	8	12,265	10.72	5,581
BaiduQA	35	179,042	4.11	26,560

As mentioned in Section 3.2, a parameter ϵ is required to determine the semantic relatedness knowledge provided by the auxiliary word embeddings. We employ a simple manual process to decide the value of ϵ. The value of ϵ is determined by manually examining some randomly sampled words. More specifically, a random set of 100 words are selected from the word embeddings, then the top-100 most related words to each of the selected words are computed based on the cosine similarity of their vector representations *i.e.,* $sr(w, w')$. These semantically related words are printed along with their cosine similarity scores to the selected word in descending order. Then we manually choose a reasonable ϵ after examining these related words. Based on the values, we set ϵ to 0.5 and 0.7 respectively for GPU-DMM over the Snippet and BaiduQA datasets. We argue that the optimal value of ϵ depends on both the external text corpus and the algorithm used in learning the word embeddings.

After setting ϵ, we observe that for some words, the number of semantic relatedness words (*i.e.,* the size of $\mathbb{M}_w$) is very large. For example, there are 129 words semantically related to word *houston*, including *dallas*, *altanta*, *orlando*, and *cleveland*. In fact, most of the 129 words refer to geographic areas like the word *houston* itself. Another example is word *carl*, which is highly related to many English names like *gordon*, *henry*, and *james*. Such words often refer to semantic relatedness in some specific domains (*e.g.,* geographical, temporal, or person names) rather than general concepts as the examples listed in Table 1. Hence, in our experiments, we simply disregard all the related words in $\mathbb{M}_w$ if $|\mathbb{M}_w| > 20$ [8].

As shown in the model overview in Figure 1, the preparation of word embeddings and $\mathbb{M}$ is independent of the proposed GPU-DMM model. The preparation can be done off-line.

Methods and Parameter Setting. We compare our GPU-DMM against the following four state-of-the-art topic models specific to short texts. For all the methods in comparison, we set the hyper-parameters $\alpha = 50/K$ and $\beta = 0.01$ unless explicitly specified elsewhere.

- **Biterm Topic Model (BTM)** learns the topics by directly modeling the generation of word co-occurrence patterns in the short text corpus [39]. In BTM, a *biterm* is an unordered word pair co-occurred in a short context.

- **Self-Aggregation based Topic Model (SATM)** assumes that each short text is sampled from a long pseudo-document unobserved in the current text collection [30]. It requires to set the number of pseudo-documents as a parameter. We tune the number of pseudo-documents from 100 to 1000 with a step of 100 in terms of classification accuracy over the two datasets (See Section 4.4) and set it to 200 and 700 respec-

[3] Our implementation is available at https://github.com/NobodyWHU/GPUDMM

[4] http://zhidao.baidu.com

[5] Stop word list is from NLTK: http://www.nltk.org/

[6] https://code.google.com/p/word2vec

[7] http://baike.baidu.com/

[8] The advanced filtering strategy is deferred to our future work.

tively, on the Snippet and BaiduQA datasets, for achieving the best classification results.

- **Dirichlet Multinomial Mixture (DMM)** assumes that each short document has only one topic, and works as the basis of our model [40].

- **Latent Feature model with DMM (LF-DMM)** integrates word embeddings into DMM by replacing the topic-word Dirichlet multinomial component with a mixture of two components: a Dirichlet multinomial component, and a word embedding component [24]. A hard constraint in LF-DMM is that all words in the short text corpus should have corresponding word embeddings. Therefore we remove all the words without the related word embeddings. We use the implementation provided by the authors and use the recommended settings with $\lambda = 0.6$, $\alpha = 0.1$, $\beta = 0.01$ as in their paper.[9]

As for GPU-DMM, we need to set the amount of promotion μ for each semantically related word. We empirically set μ to be 0.1 and 0.3 respectively for BaiduQA and Snippet datasets.

Note that [11] also proposes a LDA-based model by using multivariate Gaussian distributions with auxiliary word embeddings. Their model assumes a Multinomial topic distribution for each document, which has been proven to be inappropriate for short texts [13, 30, 42]. Hence, we exclude this model from the comparison.

We run Gibbs sampling for $1,000$ iterations and report the average results over 5 runs for all methods. The only exception is that $2,000$ iterations ($1,500$ iterations with baseline model + 500 iterations with LF-DMM) are run for LF-DMM, as in its original paper [24]. We evaluate the performance of all models in terms of topic coherence and text classification. The statistical significance is based on the student *t-test*.

4.3 Evaluation by Topic Coherence

The topics generated by each model are evaluated by the *topic coherence* metric. Traditionally, topic models are evaluated using perplexity. However, as shown in [3], perplexity does not reflect the semantic coherence of a topic. It can sometimes be contrary to human judgments. Topic coherence measures the extent that the most probable words of a topic tend to co-occur together within the same documents. It has been shown to be a better metric to assess topic quality [21].

Following [8], we use the PMI-Score proposed in [23] to calculate topic coherence. Given a topic k and its top T words with highest probabilities $(w_1, \ldots, w_T)$, the PMI-Score of k is:

$$PMI(k) = \frac{2}{T(T-1)} \sum_{1 \le i < j \le T} \log \frac{p(w_i, w_j)}{p(w_i)p(w_j)} \qquad (9)$$

In this equation, $p(w_i)$ is the probability that word w_i appears in a document, and $p(w_i, w_j)$ is the probability that words w_i and w_j appear in the same document. The overall topic coherence for each model is the averaged PMI-Score over all learnt topics. A higher topic coherence indicates the better learnt topics. Note that an external corpus is needed to calculate the PMI-Scores in Equation 9. In our experiments, we use 3 million English Wikipedia articles and 7 million Chinese Baike articles for the Snippet and BaiduQA datasets, respectively.

Figures 2 and 3 report the topic coherence of all models on the two datasets with number of top words per topic $T = \{5, 10, 20\}$ and number of topics $K = \{40, 60, 80\}$, respectively. On the Snippet dataset, GPU-DMM achieves the best topic coherence across all settings, and the improvement over other baseline models are statistical significance at the 0.01 level. BTM is the second best model in most cases, and always outperforms DMM and LF-DMM. On BaiduQA dataset, LF-DMM outperforms all other models in 6 out of 9 settings. GPU-DMM achieves the best performance in the rest 3 settings, and performs the second best overall. On both datasets, GPU-DMM significantly outperforms DMM at the 0.01 level.

Recall that the BaiduQA dataset was preprocessed with Chinese word segmentation by the dataset provider [8, 39]. For learning word embeddings and evaluating topic coherence, we used ICTCLAS[10], a Chinese word segmentation tool, to segment the words in the 7 million Chinese articles. The differences in Chinese word segmentation might introduce some performance variations.

4.4 Evaluation by Short Text Classification

With topic modeling, we can represent each document with its topic distribution $p(z|d)$. Hence, the quality of the topics can be assessed by the accuracy of text classification using the topic-level representation, as an indirect evaluation. A better classification accuracy means that the learnt topics are more discriminative and representative. Here, we employ a linear kernel Support Vector Machines (SVM) classifier in sklearn[11] with default parameter settings. The classification accuracy is computed through 5-fold cross validation on both datasets.

Methods for Topic-level Representation. Three models DMM, LF-DMM and GPU-DMM all assume only one topic for each short document. Estimating the topic distribution directly based on the topic assignment in the last Gibbs samplings for these models is inappropriate. It is reported that the post inference of document representation $p(z|d)$ based on $p(w|z)$ and $p(z)$ is important for the classification accuracy [30]. There are two ways to infer $p(z|d)$:

- Naïve Bayes (NB) rule:

$$p(z = k|d) \propto p(z = k) \prod_{i=1}^{N_d} p(w_i|z = k)$$

- Summation over words (SW):

$$p(z = k|d) \propto \sum_{w} p(z = k|w)p(w|d)$$

where $p(w|d)$ is estimated based on the relative frequency of w in d.

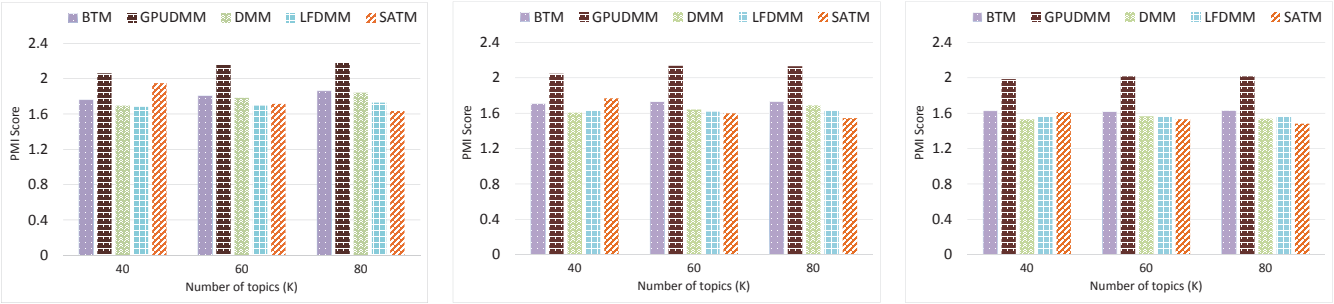
Both the NB and SW representations have been used in earlier studies. BTM uses a variant of SW post inference by replacing w with *biterm* b [39]. However, in their comparison, they use NB method for DMM instead. SW method is used for DMM and LDA and proven to largely improve the classification accuracy in [30]. However, no studies have conducted a thorough comparison of the two methods.

Here, we first compare the two document representation methods by using DMM, GPU-DMM, and BTM on Snippet dataset. Table 3 reports the classification accuracy of using the two methods with different settings on the number of topics K=$\{40, 60, 80\}$. Observe that using SW method leads to large performance improvement over NB method in topic-level representation of short documents, regardless the number of topics or the underlying topic model. Table 3 also shows that GPU-DMM achieves the best classification accuracy over all settings, followed by DMM. Similar observations hold on BaiduQA dataset (results not shown).

[9] Authors' implementation is available at https://github.com/datquocnguyen/LFTM

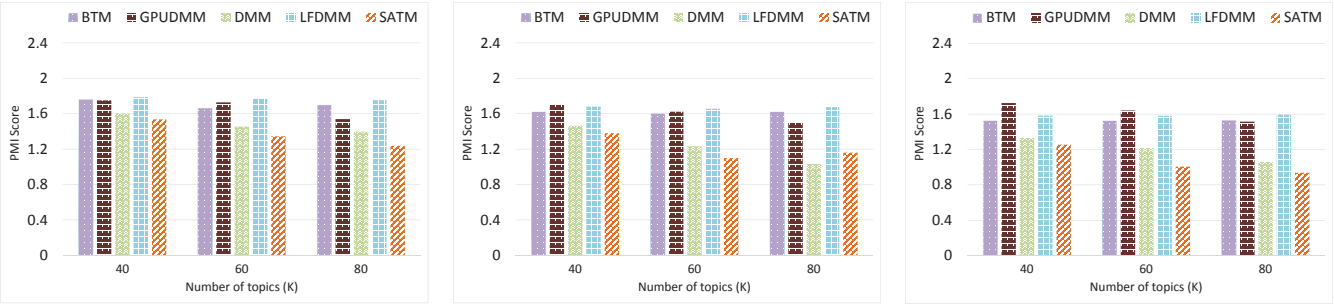
[10] http://ictclas.nlpir.org

[11] http://scikit-learn.org/



(a) Top-5 topic words (b) Top-10 topic words (c) Top-20 topic words

Figure 2: Topic Coherence on Snippet Dataset



(a) Top-5 topic words (b) Top-10 topic words (c) Top-20 topic words

Figure 3: Topic Coherence on BaiduQA Dataset

Table 3: Classification accuracy of two post inference methods (NB and SW) on Snippet dataset. The best results are highlighted in boldface for each K setting.

Model	Method	K = 40	K = 60	K = 80
DMM	NB	0.7180	0.7590	0.7470
	SW	0.8298	0.8522	0.8479
GPU-DMM	NB	0.7510	0.7600	0.7650
	SW	**0.8539**	**0.8667**	**0.8722**
BTM	NB	0.7100	0.7120	0.6700
	SW	0.8100	0.8183	0.8272

Table 4: The top 3 topics and their proportions in an example document by using DMM.

Top	SW		NB	
	Topic Id	$p(z\|d)$	*Topic Id*	$p(z\|d)$
1	37	0.3187	37	0.9999
2	36	0.0820	27	2.92e-09
3	27	0.0760	36	4.57e-10

To better understand the big difference in classification accuracy by using the two different representation methods NB and SW, we show the top 3 topics with the corresponding $p(z|d)$ values for a random document in Snippet dataset by the two methods in Table 4. Observe that the top-3 topics for the document are the same (*i.e.,* topic Ids 27, 36, 37) inferred by both methods NB and SW. The relative proportions of each topic inferred by the two methods are tremendously different, particularly for the second and third topics. Using NB method, top-1 topic dominates the topic representation.

Table 5: Average classification accuracy of the 5 models on two datasets, with different number of topic K settings. The best results are highlighted in boldface on each dataset. † indicates that the difference to the best result is statistically significant at 0.01 level.

Dataset	Model	$K = 40$	$K = 60$	$K = 80$
Snippet	BTM	$0.8100^{\dagger}$	$0.8183^{\dagger}$	$0.8272^{\dagger}$
	DMM	$0.8298^{\dagger}$	$0.8522^{\dagger}$	$0.8479^{\dagger}$
	SATM	$0.8284^{\dagger}$	$0.8231^{\dagger}$	$0.8235^{\dagger}$
	LF-DMM	$0.7414^{\dagger}$	$0.7409^{\dagger}$	$0.7462^{\dagger}$
	GPU-DMM	**0.8539**	**0.8667**	**0.8722**
BaiduQA	BTM	$0.5098^{\dagger}$	$0.5336^{\dagger}$	$0.5486^{\dagger}$
	DMM	$0.5228^{\dagger}$	$0.5476^{\dagger}$	$0.5532^{\dagger}$
	SATM	$0.4872^{\dagger}$	$0.4567^{\dagger}$	$0.4605^{\dagger}$
	LF-DMM	$0.4240^{\dagger}$	$0.4486^{\dagger}$	$0.4868^{\dagger}$
	GPU-DMM	**0.5439**	**0.5637**	**0.5708**

Because NB method calculates $p(z|d)$ by a series of product of $p(w|z)$ over w, some irrelevant words could incur too much penalty for topic z. It was well studied that Naive Bayes makes unrealistic independence assumptions, push probabilities towards 0 and 1 [25]. As the result, NB method leads to extremely sparse topic distribution, equivalent to using the direct topic assignment. This is not a desired method for short texts. In the rest of this paper, we only report the classification accuracies by using the SW method.

Classification Accuracy. Table 5 reports the document classification accuracy on the two datasets by using the 5 models in comparison. We make the following observations.

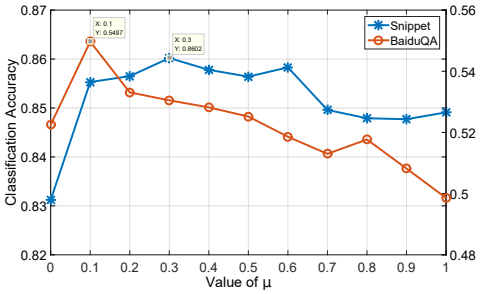


Figure 4: Effect of weight μ under $K = 40$

First, our GPU-DMM model significantly outperforms other state-of-the-art models on both datasets. Particularly, the significant performance gains of GPU-DMM with respect to DMM are achieved on all $K = \{40, 60, 80\}$ settings. This validates that incorporating general word semantic relatedness knowledge based on word embeddings with GPU models is beneficial for short text topic modeling.

Second, in our experiments, DMM is the second best performing model in the document classification task. Although BTM was shown to perform better than DMM in [39], as we discussed earlier, the NB based document representation was used for DMM in [39], which may be the reason for the poorer performance. By using the SW inference method, we show that BTM is inferior to DMM in this classification task on both datasets. This result suggests that generating biterms may bring little discriminative word co-occurrence knowledge, which is also observed in [30, 41].

Third, SATM delivers mixed performance on the two datasets. On Snippet dataset, it performs slightly better than BTM when $K = 40$ and $K = 60$; it performs much poorer than BTM on BaiduQA dataset. Surprisingly, we observe that LF-DMM performs the worst among all models on both datasets even though it also exploits word embeddings. One possible reason for its modest performance is that harnessing the semantic relatedness via a two-component mixture may need a more complex mechanism. A simple switch mechanism with an indication variable used in LF-DMM may not optimally balance the two components.

Effect of the Promotion Weight μ. We now study the effect of weight μ in GPU-DMM (see Equation 3). Figure 4 depicts the classification accuracy of different μ settings under $K = 40$ on the two datasets. Note that, when $\mu = 0$, GPU-DMM is equivalent to the DMM model.

While GPU-DMM obtains significant improvements over DMM (*i.e.,* $\mu = 0$) with any positive μ values on Snippet dataset, it only achieves positive gains when μ is relatively small on BaiduQA dataset, *i.e.,* $\mu \leq 0.5$. Specifically, on BaiduQA dataset, the best accuracy is achieved when $\mu = 0.1$, further increasing μ results in performance degradation. Reported in Table 2, documents in BaiduQA is much shorter than that in Snippet dataset. Because of the very limited context information in such short document (4.11 words per document on average), that chance of sampling an incorrect topic for a document becomes high. Setting a larger μ means more related words in the wrongly sampled topic are promoted, leading to poorer results.

4.5 Efficiency

In the last set of experiments, we compare the running time of the models. We implement GPU-DMM, DMM, SATM and BTM models in Java, and use the Java implementation provided by its authors for LF-DMM.

Table 6: Time cost (in seconds) per iteration of each model on two datasets, with different settings on number of topics $K = \{40, 60, 80\}$. The best and the second best results are highlighted in boldface and underlined respectively.

Dataset	Model	$K = 40$	$K = 60$	$K = 80$
Snippet	BTM	0.572	0.832	1.121
	DMM	**0.042**	**0.069**	**0.088**
	SATM	0.392	0.451	0.478
	LF-DMM	5.209	7.234	9.702
	GPU-DMM	<u>0.085</u>	<u>0.115</u>	<u>0.217</u>
BaiduQA	BTM	<u>1.411</u>	2.042	2.603
	DMM	**0.355**	**0.566**	**0.843**
	SATM	10.120	10.345	11.711
	LF-DMM	13.028	20.493	31.822
	GPU-DMM	1.558	<u>1.862</u>	<u>2.318</u>

Table 6 reports the average run-time (in seconds) per iteration for each model. The simplest model DMM, is the most efficient one as expected. GPU-DMM is slightly slower than DMM, being the second most efficient model. This result is expected because GPU-DMM shares similar Gibbs sampling process with DMM. The extension of the sampling process by taking in semantically related words through GPU does not cost much in terms of computation. The average run-time of GPU-DMM is less than 3 times of DMM for different topic numbers K on both datasets. This observation is consistent with the analysis made in Section 3.3.

SATM is much faster than BTM on Snippet dataset, but requires much more time than BTM on BaiduQA dataset. As being described in [30], the computational cost of SATM is positively proportional to the number of pseudo-documents. Hence, the large difference in the run-time on the two datasets is due to the setting of the optimal number of pseudo-documents, *i.e.,* 200 and 700 respectively on the two datasets. LF-DMM is the slowest model in this comparison. As discussed in Section 2, LF-DMM estimates the word embedding component for each topic by projecting the topic into the vector space, where a regularized log-linear has to be optimized. This process is time-consuming.

In summary, our earlier results and the results in Table 6 suggest that GPU-DMM is a desired choice for short text topic modeling, with respect to both effectiveness and efficiency.

5. CONCLUSION

Unlike normal documents, short texts carry limited context information, causing severe sparsity problems when applying conventional topic models. In this paper, we propose a new topic model, named GPU-DMM, to leverage global word co-occurrence knowledge to help distil better topics over short texts. Instead of extracting word semantic relatedness knowledge from external thesauri, we propose to harness the knowledge from the results of the recent neural network language model techniques. GPU-DMM enhances the topic similarity for two semantically related words which rarely co-occur in short texts. We conduct extensive experiments on two real-world short text corpora. The experimental results show that GPU-DMM outperforms existing state-of-the-art alternatives in terms of effectiveness and efficiency. Nevertheless, there is still room to improve our model in the future. For example, we would like to adjust the promotion weight μ based on the topic and correlated word pair together, which is a fixed value in this work. Moreover, it is interesting to validate the effectiveness of using other word embedding techniques like Glove [27].

Acknowledgements. This research was supported by National Natural Science Foundation of China (No. 61502344), Natural Science Foundation of Hubei Province (No. 2015CFB337), Natural Scientific Research Program of Wuhan University (No. 2042015kf0014), and Singapore Ministry of Education Academic Research Fund Tier 2 (MOE2014-T2-2-066). Chenliang Li is the corresponding author.

6. REFERENCES

[1] Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, and J.-L. Gauvain. *Neural probabilistic language models*. Springer, 2006.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 2003.

[3] J. Chang, S. Gerrish, C. Wang, J. L. Boyd-Graber, and D. M. Blei. Reading tea leaves: How humans interpret topic models. In *NIPS*, 2009.

[4] M. Chen, X. Jin, and D. Shen. Short text classification improved by learning multi-granularity topics. In *IJCAI*, 2011.

[5] Z. Chen and B. Liu. Mining topics in documents: standing on the shoulders of big data. In *SIGKDD*, 2014.

[6] Z. Chen, A. Mukherjee, B. Liu, M. Hsu, M. Castellanos, and R. Ghosh. Discovering coherent topics using general knowledge. In *CIKM*, 2013.

[7] Z. Chen, A. Mukherjee, B. Liu, M. Hsu, M. Castellanos, and R. Ghosh. Leveraging multi-domain prior knowledge in topic models. In *IJCAI*, 2013.

[8] X. Cheng, X. Yan, Y. Lan, and J. Guo. BTM: topic modeling over short texts. *IEEE Trans. Knowl. Data Eng.*, 2014.

[9] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, 2008.

[10] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 2011.

[11] R. Das, M. Zaheer, and C. Dyer. Gaussian LDA for topic models with word embeddings. In *ACL*, 2015.

[12] T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, 1999.

[13] L. Hong and B. D. Davison. Empirical study of topic modeling in twitter. In *The First Workshop on Social Media Analytics*, 2010.

[14] L. Hong, D. Yin, J. Guo, and B. D. Davison. Tracking trends: incorporating term volume into temporal topic models. In *SIGKDD*, 2011.

[15] O. Jin, N. N. Liu, K. Zhao, Y. Yu, and Q. Yang. Transferring topical knowledge from auxiliary long texts for short text clustering. In *CIKM*, 2011.

[16] T. Kenter and M. de Rijke. Short text similarity with word embeddings. In *CIKM*, 2015.

[17] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger. From word embeddings to document distances. In *ICML*, 2015.

[18] H. Mahmoud. Polya urn models. *Chapman & Hall/CRC Texts in Statistical Science*, 2008.

[19] R. Mehrotra, S. Sanner, W. Buntine, and L. Xie. Improving lda topic models for microblogs via tweet pooling and automatic labeling. In *SIGIR*, 2013.

[20] T. Mikolov, K. Chen, G. Corrada, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[21] D. Mimno, H. M. Wallach, E. Talley, M. Leenders, and A. McCallum. Optimizing semantic coherence in topic models. In *EMNLP*, 2011.

[22] A. Mnih and G. E. Hinton. A scalable hierarchical distributed language model. In *NIPS*, 2009.

[23] D. Newman, J. H. Lau, K. Grieser, and T. Baldwin. Automatic evaluation of topic coherence. In *HLT-NAACL*, 2010.

[24] D. Q. Nguyen, R. Billingsley, L. Du, and M. Johnson. Improving topic models with latent feature word representations. *TACL*, 2015.

[25] A. Niculescu-Mizil and R. Caruana. Predicting good probabilities with supervised learning. In *ICML*.

[26] K. Nigam, A. K. MacCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 2000.

[27] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.

[28] X.-H. Phan, L.-M. Nguyen, and S. Horiguchi. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *WWW*, 2008.

[29] I. Porteous, D. Newman, A. Ihler, A. Asuncion, P. Smyth, and M. Welling. Fast collapsed gibbs sampling for latent dirichlet allocation. In *SIGKDD*, 2008.

[30] X. Quan, C. Kit, Y. Ge, and S. J. Pan. Short and sparse text topic modeling via self-aggregation. In *AAAI*, 2015.

[31] Z. Ma, A. Sun, Q. Yuan, and G. Cong. Topic-driven reader comments summarization. In *CIKM*, 2012.

[32] D. Ramage, S. T. Dumais, and D. J. Liebling. Characterizing microblogs with topic models. In *ICWSM*, 2010.

[33] D. E. Rumelhar, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Cognitive Modeling*, 1988.

[34] B. Sriram, D. Fuhry, E. Demir, H. Ferhatosmanoglu, and M. Demirbas. Short text classification in twitter to improve information filtering. In *SIGIR*, 2010.

[35] A. Sun. Short text classification using very few words. In *SIGIR*, 2012.

[36] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In *SIGKDD*, 2011.

[37] X. Wang, C. Zhai, X. Hu, and R. Sproat. Mining correlated bursty topic patterns from coordinated text streams. In *SIGKDD*, 2007.

[38] J. Weng, E.-P. Lim, J. Jiang, and Q. He. Twitterrank: finding topic-sensitive influential twitterers. In *WSDM*, 2010.

[39] X. Yan, J. Guo, Y. Lan, and X. Chen. A biterm topic model for short texts. In *WWW*, 2013.

[40] J. Yin and J. Wang. A dirichlet multinomial mixture model-based approach for short text clustering. In *SIGKDD*, 2014.

[41] X. Yunqing, T. Nan, H. Amir, and C. Erik. Discriminative bi-term topic model for headline-based social news clustering. In *AAAI*, 2015.

[42] W. X. Zhao, J. Jiang, J. Weng, J. He, E.-P. Lim, H. Yan, and X. Li. Comparing twitter and traditional media using topic models. In *ECIR*, 2011.

[43] G. Zheng and J. Callan. Learning to reweight terms with distributed representations. In *SIGIR*, 2015.