# Enhancing Topic Modeling for Short Texts with Auxiliary Word Embeddings

CHENLIANG LI, Wuhan University
YU DUAN, Wuhan University
HAORAN WANG, Wuhan University
ZHIQIAN ZHANG, Wuhan University
AIXIN SUN, Nanyang Technological University
ZONGYANG MA, Nanyang Technological University

Many applications require semantic understanding of short texts, and inferring discriminative and coherent latent topics is a critical and fundamental task in these applications. 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. Recent studies show that the Dirichlet Multinomial Mixture (DMM) model is effective for topic inference over short texts by assuming that each piece of short text is generated by a single topic. However, DMM has two main limitations. First, even though it seems reasonable to assume that each short text has only one topic because of its shortness, the definition of "shortness" is subjective and the length of the short texts is dataset dependent. That is, the single topic assumption may be too strong for some datasets. To address this limitation, we propose to model the topic number as a Poisson distribution, allowing each short text to be associated with a small number of topics (*e.g.,* 1 to 3 topics). This model is named PDMM. Second, DMM (and also PDMM) does not have access to background knowledge (*e.g.,* semantic relations between words) when modeling short texts. When a human being interprets a piece of short text, the understanding is not solely based on its content words, but also their semantic relations. Recent advances in word embeddings offer effective learning of word semantic relations from a large corpus. Such auxiliary word embeddings enable us to address the second limitation. To this end, we propose to promote the semantically related words under the same topic during the sampling process, by using the generalized Pólya urn (GPU) model. Through the GPU model, background knowledge about word semantic relations learned from millions of external documents can be easily exploited to improve topic modeling for short texts. By directly extending the PDMM model with the GPU model, we propose two more effective topic models for short texts, named GPU-DMM and GPU-PDMM. Through extensive experiments on two real-world short text collections in two languages, we demonstrate that PDMM achieves better topic representations than state-of-the-art models, measured by topic coherence. The learned topic representation leads to better accuracy in a text classification task, as an indirect evaluation. Both GPU-DMM and GPU-PDMM further improve topic coherence and text classification accuracy. GPU-PDMM outperforms GPU-DMM at the price of higher computational costs.

General Terms: Algorithms, Management, Experimentation

Additional Key Words and Phrases: Topic Model, Short Texts, Word Embeddings

**ACM Reference Format:**
Chenliang Li, Yu Duan, Haoran Wang, Zhiqian Zhang, Aixin Sun, and Zongyang Ma, 2017. Enhancing Topic Modeling for Short Texts with Auxiliary Word Embeddings. *ACM Trans. Inf. Syst.* V, N, Article A (January YYYY), 31 pages.
DOI: 0000001.0000001

This paper is an extended version of the paper [Li et al. 2016] presented at the 39th International ACM SIGIR conference (Pisa, Italy, July 17-21, 2016).

Author's addresses: C. Li (corresponding author), Y. Duan, H. Wang and Z. Zhang, State Key Lab of Software Engineering, Computer School, Wuhan University, China 430072; A. Sun and Z. Ma, School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798.

DOI: 0000001.0000001

## 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 question/answer pairs, to name a few. Given the large volume of short texts available, effective and efficient models to discover the latent topics from short texts becomes fundamental to many applications that require semantic understanding of textual content, such as user interest profiling [Weng et al. 2010], topic detection [Wang et al. 2007], comment summarization [Ma et al. 2012], content characterizing [Ramage et al. 2010], and classification [Sriram et al. 2010].

Conventional topic modeling techniques, *e.g.,* pLSA and LDA, are widely used to infer latent topical structure from text corpus [Blei et al. 2003; Hofmann 1999]. 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 the word distribution of each topic, based on higher-order word co-occurrence patterns [Porteous et al. 2008]. These models and their variants have been studied extensively for various tasks in information retrieval and text mining [Hong et al. 2011; Wang and Blei 2011; Zhao et al. 2011]. 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 semantically less coherent.

Several 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 [Hong and Davison 2010; Mehrotra et al. 2013; Weng et al. 2010]. 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 [Nigam et al. 2000; Yin and Wang 2014; Zhao et al. 2011]. 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 [Yan et al. 2013; Zhao et al. 2011]. However, this single topic assumption may be too strong in reality due to the nature of the short text collection. Many short texts could cover more than one topic (see Table I for some examples). 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 [Yan et al. 2013] and using soft-clustering mechanism for augmenting word co-occurrence [Quan et al. 2015]. However, the word co-occurrence information that can be captured by these models is limited to the short text corpus itself. That is, these models cannot fully capture words that are semantically related but rarely co-occur in the given short text corpus.

When a human being interprets a piece of text, the understanding is not solely based on its content, but also on her background knowledge, *e.g.,* semantic relations between words. Hence, it is natural to exploit external lexical knowledge to guide the topic inference over short texts. Existing work in this line largely relies on either external thesauri (*e.g.,* WordNet) or lexical knowledge derived from documents in a specific domain (*e.g.,* product comments) [Chen and Liu 2014; Chen et al. 2013a,b]. 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.

Effective learning of general word semantic relations is now feasible and practical with recent developments in neural network techniques, which have contributed improvements in many tasks in Information Retrieval (IR) and Natural Language Processing (NLP) [Collobert and Weston 2008; Collobert et al. 2011; Kenter and de Rijke 2015; Kusner et al. 2015; Zheng and Callan 2015]. Specifically, neural network language models, *e.g.,* Continuous Bag-of-Words (CBOW), Continuous Skip-gram model, and Glove model [Mikolov et al. 2013; Pennington et al. 2014], learn word embeddings (or word vectors) with the aim of fully retaining the contextual information for each word, including both semantic and syntactic relations. Such general word semantic relations 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, publicly available on the Web.[1]

Because of its good performance, in this paper, we propose to extend the DMM model for topic modeling over short texts by addressing its two limitations.

- To address the limitation of single topic assumption, we propose a Poisson-based Dirichlet Multinomial Mixture model for short texts, named **PDMM**. PDMM is a variant of DMM that allows each short text to be generated by one or more (but not too many) topics. Specifically, each short text in PDMM is assumed to be generated by a limited number of topics (*e.g.,* 1, 2, or 3 topics), where the topic number is modeled as a Poisson distribution.
- To address the limitation of DMM (and also PDMM) having no access to background knowledge external to the given short text corpus, we propose to exploit the general word semantic relations during the topic inference process. The general word semantic relations are encoded in word embeddings, learned from millions of external documents (*e.g.,* Wikpedia). The incorporation of the auxiliary word embeddings is through the *generalized Pólya urn* (GPU) model [Mahmoud 2008] during the sampling process, which effectively tackles the data sparsity issue. Extended with the GPU model, we propose two new models, namely **GPU-DMM** and **GPU-PDMM**, which extend the DMM and PDMM models, respectively.

Figure 1 gives an overview of the proposed GPU-PDMM model. The model promotes semantically relevant words under the same topic after sampling a topic for a short document. In this sense, the semantically relevant words are linked 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 to guide the topic inference process, such that only appropriate external knowledge is exploited for the sampled topic. The same overview applies to GPU-DMM. Note that two topics are discovered in the short document illustrated in Figure 1. If GPU-DMM is applied, only a single topic will be assigned to the short document in Figure 1, because of the underlying assumption of DMM model. Both models are *fast* and *flexible* because the word embeddings can be pre-learned from any other large text collections. On two real-world datasets in two languages (*i.e.,* snippets of an English search engine, and questions from a Q&A service in Chinese), the proposed models, PDMM, GPU-DMM, and GPU-PDMM, discover more prominent topics than state-of-the-art alternatives. The learned topic representations also achieve better classification accuracy than baseline models. The main contributions of this paper are summarized as follows:

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

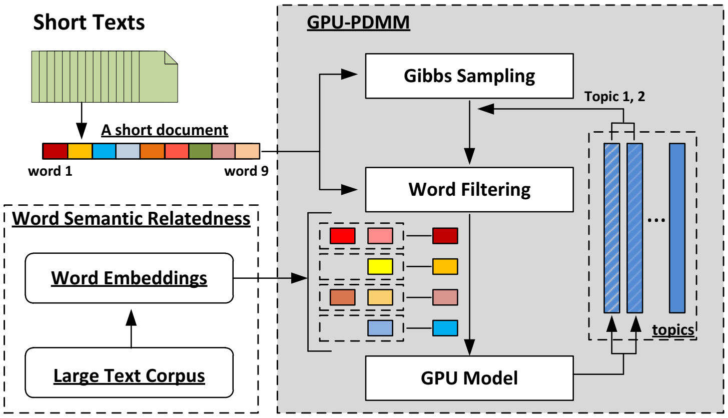


Fig. 1. Overview of the proposed GPU-PDMM model.

(1) We propose an effective Poisson-based Dirichlet Multinomial Mixture model to relax the single topic assumption of DMM. In PDMM each short text is sampled from a small number of topics (*e.g.,* 1 to 3 topics). To the best of our knowledge, this is the first model to provide a mechanism to constrain topic number distribution in the modeling process.

(2) We introduce a simple, fast, and effective mechanism to incorporate word semantic relations in the latent topic learning over short texts. The word semantic relations, encoded in the form of word embeddings, are learned from a large text corpus, which is domain independent and ease to access. To the best of our knowledge, this is the first work for a topic model to exploit word embeddings with the GPU model.

(3) On two real-world short text collections in two languages, we evaluate the proposed PDMM, GPU-DMM and GPU-PDMM models against 8 state-of-the-art alternative models. Experimental results demonstrate the effectiveness of the proposed models, measured by topic coherence and text classification accuracy.

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

## 2. RELATED WORK

We review recent advances on learning better topic representations on short texts. We then focus on 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. [2008] propose to infer topic structure of short texts by using the learnt latent topics from Wikipedia. Similarly, Jin et al. [2011] infer latent topics of short texts for clustering by using auxiliary long texts. 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 [Weng et al. 2010], 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 [Hong and Davison 2010; Mehrotra et al. 2013; Zhao et al. 2011]. 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.

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 [Yin and Wang 2014; Zhao et al. 2011]. DMM is based on the assumption made in the mixture of unigrams model proposed by Nigam et al. [2000], *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 [Quan et al. 2015; Yan et al. 2013; Zhao et al. 2011]. Yin and Wang [2014] propose a collapsed Gibbs Sampling algorithm for DMM and show its effectiveness in short text clustering. Due to its simplicity and effectiveness, we develop PDMM on the basis of DMM, as its name suggests. One critical defect of DMM is that associating all words of a short document to a single topic could introduce much noise. In this work, we devise the PDMM model to allow a short document to be related with a limited number of relevant topics, leading to better discriminative and coherent topics.

Recently, many efforts have been spent towards intensifying the word co-occurrence information from the collection of short texts being modeled. Yan et al. [2013] 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. Their experimental results show that BTM produces discriminative topic representations as well as more coherent topics for short texts. Zuo et al. [2016b] propose a novel word co-occurrent network based topic model (WNTM) for short text. Instead of learning the topic distribution of each short document, WNTM directly infers the topic distribution of a word by using all the words co-occurring with it. Since the word co-occurrences are adequate, the sparsity and imbalance problem of short texts is therefore mitigated. Inspired by the aforementioned aggregation strategies, Quan et al. [2015] propose a self-aggregation based topic model (SATM) for short texts. 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. Motivated by the promising potential of SATM, Zuo et al. [2016a] revise SATM into an efficient pseudo-document-based topic model (PTM) for short text topic modeling. Similar to SATM, PTM assumes that a short text is a part of a long pseudo-document. However, as a contrast, a short document belongs to a single pseudo-document only in PTM, which results in a more efficient inference process than SATM.

**Topic Models for Short Texts with Word Embeddings.** Word embeddings, first introduced in [Rumelhart et al. 1988], have been successfully applied in language models and many NLP tasks, such as named entity recognition and parsing [Bengio et al. 2006; Mnih and Hinton 2009]. 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. [2015]. 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 computationally expensive. Several works aim to model topics by using multivariate Gaussian distributions with word embeddings. Das et al. [2015] proposes an LDA-based topic model by using multivariate Gaussian distributions with word embeddings. Sridhar [2015] propose a Gaussian mixture topic model (GMTM) to infer the topic structure by using word embeddings. In GMTM, each learnt Gaussian component refers to a latent topic. And the word embeddings are learnt based on the document corpus itself. Similarly, Hu and Tsujii [2016] propose a latent concept topic model (LCTM) by modeling a concept as a Gaussian distribution in the continuous word embedding space. In contrast to GMTM, the word embeddings learnt from a large external corpus are used in LCTM. In our earlier work [Li et al. 2016], we extend DMM model by exploiting the general word semantic relations provided by auxiliary word embeddings by using the generalized Pólya urn (GPU) model in the topic inference of short texts, named GPU-DMM. In GPU-DMM, GPU model is utilized to promote the semantically related words under the same topic. This mechanism leads to better topic coherence and more discriminative topic representation for the short texts. Here, we further extend GPU-DMM model by combining GPU and PDMM, leading to the GPU-PDMM model. In contrast to GPU-DMM, GPU-PDMM relaxes the single topic assumption such that a short document can be associated with more than one relevant topic. Compared with existing approaches of incorporating word embeddings in topic models, GPU reduces the computational cost significantly. To the best of our knowledge, both GPU-DMM and GPU-PDMM are the first attempt to combine word embeddings and GPU model for solving sparsity problem of short texts.

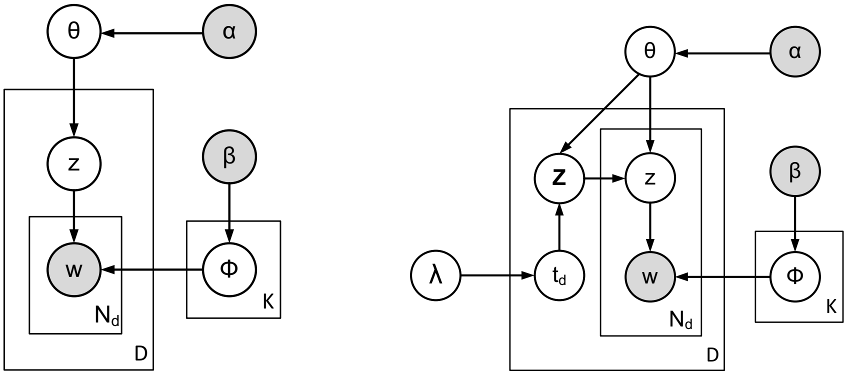
## 3. POISSON-BASED DIRICHLET MIXTURE MODEL

As its name suggests, the proposed PDMM is an extended DMM model. Given a short document, PDMM first samples a topic number for it based on a Poisson distribution. The specific topics are then sampled based on global topic distribution as well as the related topic-word distributions. Next, we review the DMM model and detail the proposed PDMM.

### 3.1. Dirichlet Mixture Model

The Dirichlet Mixture Model is a generative probabilistic model with the assumption that *a document is generated from a single topic* [Nigam et al. 2000; Yin and Wang 2014]. That is, all the words within a document are generated by the same topic distribution.

Given a short text corpus of $D$ documents, with a vocabulary of size $V$, and $K$ predefined latent topics, each document $d$ is associated with one specific topic $k$. Then the $N_d$ words $\{w_{d,1}, w_{d,2}, \ldots, w_{d,N_d}\}$ in document $d$ are generated by the topic-word multinomial distribution $p(w|z = k) = \phi_k$ by assuming independence of the words



(a) DMM

(b) PDMM

Fig. 2. Graphical representation of (a) DMM, and (b) PDMM.

themselves. More formally, with Dirichlet priors $\alpha$ and $\beta$, the generative process of DMM is described as follows:

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

The graphical representation of DMM is shown in Figure 2(a). The hidden variables in the generative process can be approximated by applying Gibbs sampling. Following the approach in [Yin and Wang 2014], a topic $z$ is sampled for each document in every iteration according to the following conditional distribution:

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

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:

$$p(w|z = k) = \frac{n_k^w + \beta}{\sum_v^V n_k^v + V\beta} \tag{2}$$

### 3.2. Poisson-based Dirichlet Mixture Model

DMM makes an inflexible restriction that only a single topic is associated with a short text. Although this restriction could successfully alleviate the word co-occurrence sparsity problem underlying the short texts to some extent, the restriction is not always held in reality. Table I lists $5$ example short texts with their subscripts indicating the associated topics.[2] For example, in the first text $t_1$, "website$_2$ information$_2$ medical$_1$

[2] The example short texts are sampled from the dataset used in our experiments. The topic labeling is conducted with human effort.

Table I. Example short texts; the subscript of each word indicates its topic.

| Text Id | Words and their topics |
|---|---|
| $t_1$ | website$_2$ information$_2$ medical$_1$ com$_2$ healthcare$_1$ |
| $t_2$ | wiki$_1$ styles$_2$ bass$_2$ keyboard$_2$ wikipedia$_1$ guitar$_2$ music$_2$ rock$_2$ encyclopedia$_1$ prominent$_2$ |
| $t_3$ | web$_1$ search$_1$ google$_1$ educators$_2$ student$_2$ tool$_1$ research$_1$ heard$_2$ teachers$_2$ websearch$_1$ |
| $t_4$ | shtml$_1$ java$_2$ web$_1$ programming$_2$ collection$_1$ charge$_3$ books$_3$ |
| $t_5$ | nutrition$_1$ information$_2$ medical$_1$ diet$_1$ msn$_3$ health$_1$ fitness$_1$ news$_2$ msnbc$_3$ |

com$_2$ healthcare$_1$" illustrates that two topics are used to generate this text. While the words "medical, healthcare" are generated by the first topic, the rest are generated by the second topic. In texts $t_4$ and $t_5$, we can even associate $3$ topics with each text. This phenomenon is observed to be prevalent over the two text collections studied in this work (see Section 5). We argue that restricting oneself to a single topic within a short text could introduce much noise and information loss.

Here, we present a novel **P**oisson-based **D**irichlet **M**ultinomial **M**ixture model for short texts, named PDMM. Inspired by the observation made in Table I, in PDMM, we allow each short text to be generated by multiple topics. Specifically, document $d$ is associated with a topic number $t_d$, and $0 < t_d \leq \tau$ indicates the number of topics covered by this document. $\tau$ is the maximum number of topics allowable in a short text. Here, we model $t_d$ by using a Poisson distribution. The graphical representation of PDMM is shown in Figure 2(b) and the generative process 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 number $t_d \sim Poisson(\lambda)$
    (b) Sample $t_d$ distinct topics $\mathbf{Z}_d \sim Multinomial(\theta)$
    (c) For each word $w \in \{w_{d,1}, ..., w_{d,N_d}\}$
            Uniformly sample a topic $z_{d,w} \sim \mathbf{Z}_d$
            Sample a word $w \sim Multinomial(\phi_{z_{d,w}})$
```

Here, $t_d \geq 1$ is sampled from Poisson distribution with parameter $\lambda$. Since $t_d$ is a non-negative integer, we model the sampling of $t_d$ by using the rectified Poisson distribution (*i.e.,* $0 < t_d \leq \tau$). As with DMM, we utilize the Gibbs sampling to perform the approximate inference and parameter learning. The associated topic $z_{d,w}$ for each word $w$ within document $d$ is sampled as follows:

$$p(z_{d,w} = k | \vec{z}_{\neg d,w}, \mathbf{Z}_d, \vec{d}) \propto \frac{1}{t_d} \times \frac{n^w_{k,\neg(d,w)} + \beta}{\sum_v^V n^v_{k,\neg(d,w)} + V\beta} \tag{3}$$

In Equation 3, the symbol $\neg(d, w)$ means that word $w$ in document $d$ is excluded from the counting. Similarly, conditioned on all $z_{d,w}$ for document $d$ (*i.e.,* $\vec{z}_d$), we can sample each possible $\mathbf{Z}_d$ as follows:

$$p(\mathbf{Z}_d | \vec{z}_{\neg d}, \vec{z}_d, \vec{d}) \propto$$

$$\frac{\lambda^{t_d}}{t_d^{n_d}} \times \frac{\prod_{k \in \mathbf{Z}_d} (c_{k,\neg d} + \alpha)}{\prod_{i=0}^{t_d - 1} (\sum_k^K c_{k,\neg d} + K\alpha - i)} \times \prod_{k \in \mathbf{Z}_d} \frac{\prod_{w \in d_k} \prod_{i=0}^{n^w_{k,d}} (n^w_{k,\neg d} + n^w_{k,d}) - i + \beta}{\prod_{i=0}^{n_{k,d} - 1} (\sum_w^V n^w_{k,\neg d} + n_{k,d} - i + V\beta)} \tag{4}$$

where $c_k$ is the number of words associated with topic $k$, $d_k$ is the set of words that are associated with topic $k$ within document $d$, $n_{k,d}$ is the number of words that are associated with topic $k$ within document $d$, $n^w_{k,d}$ is the number of times that word $w$

is assigned to topic $k$ within document $d$. The symbol $\neg d$ means that document $d$ is excluded from the counting.

**Search Space Pruning** Given a $\tau$ value, a short text may be assigned $1 \leq t_d \leq \tau$ number of topics. As the topics are sampled from $K$ topics, then there are $\sum_{i=1}^{\tau} C_K^i$ possible $\mathbf{Z}_d$ settings in total. For example, by setting $\tau = 3$ and $K = 40$, we will have $10,700$ different possible $\mathbf{Z}_d$ settings. The huge computational costs involved in sampling $\mathbf{Z}_d$ could make PDMM impractical in real applications.

To reduce the search space of $\mathbf{Z}_d$, we propose a heuristic strategy: only sampling the topics that are more relevant to document $d$ based on the current estimation. Specifically, we infer the topic probability $p(z|d)$ under the current parameter estimation during each iteration by using Equation 5. In this equation, $p(w|d)$ is estimated based on the relative frequency of $w$ in $d$.

$$p(z = k|d) \propto \sum_{w} p(z = k|w)p(w|d) \tag{5}$$

With the values of $p(z|d)$ estimated, only the top-$M$ topics for document $d$ are used to generate $\mathbf{Z}_d$, where $\tau < M \ll K$. By limiting to the top $M$ topics, there are only $\sum_{i=1}^{\tau} C_M^i$ possible $\mathbf{Z}_d$ settings for sampling. Referring to the same example, when $\tau = 3$ and $K = 40$, by setting $M = 10$, our search space would be reduced to $175$ different $\mathbf{Z}_d$ settings, compared to the $10,700$ possible settings without setting $M$ (*i.e.,* a reduction of $98.4\%$ in search space in this case). In this work, we set $M$ to $10$ (see Section 5.5 for empirical analysis). In the rest of the paper, unless explicitly specified, PDMM refers to the model with this pruning strategy.

**Model Inference** Because all $z_{d,w}$ and $\mathbf{Z}_d$ values for document $d$ are correlated, the sampling order of $z_{d,w}$ and $\mathbf{Z}_d$ becomes crucial. Simply sampling a new $\mathbf{Z}_d$ (*i.e.,* $t_d$) based on Equation 4 with all $z_{d,w}$ values conditioned on the previous $t_d$ and $\mathbf{Z}_d$ could not converge to the true posterior distribution. This is a common issue of Markov Chain Monte Carlo (MCMC) methods, called autocorrelation phenomenon [Straatsma et al. 1986].

The Gibbs sampling process of PDMM is detailed in Algorithm 1. We firstly sample each $z_{d,w}$ within document $d$ conditioned on each possible $\mathbf{Z}_d$ by using Equation 3 (Line 23). Then, the likely $\mathbf{Z}_d$ is sampled conditioned on all the corresponding $z_{d,w}$ values by using Equation 4 (Line 24). Afterwards, all the values of $z_{d,w}$ are set to the updated $\mathbf{Z}_d$'s corresponding values sampled in the first step (Lines 25-31). The posterior distribution is also calculated by using Equation 2.

## 4. INCORPORATING AUXILIARY WORD EMBEDDINGS

Human beings can quickly capture the topics covered by a piece of short text for having background knowledge to identify the (missing) relevant information within the short text. Here, we introduce a novel mechanism based on *generalized Pólya urn* (GPU) model [Mahmoud 2008] to enhance the DMM and PDMM models, to incorporate the general word semantic relations in the topic inference process. The words that are highly relevant to the topic are then selected, and in the process, their semantically related words are extracted and promoted by using the GPU model. As shown in Figure 1, the auxiliary word embeddings utilized in GPU-PDMM is pre-learned using the state-of-the-art word embedding techniques from a large document collection. Next, we present the details of the proposed models GPU-DMM and GPU-PDMM.

```
ALGORITHM 1: PDMM
input  : Topic number $K$, $\alpha$, $\beta$, $\lambda$, $M$ and $D$ short documents
output: The posterior topic-word distribution
foreach $d \in D$ do
    $t_d \leftarrow t \sim Poisson(\lambda)$;
    sample $t_d$ distinct topics $\mathbf{Z}_d \sim Multinomial(1/K)$;
    foreach $w \in d$ do
        uniformly sample $z \sim \mathbf{Z}_d$;
        $z_{d,w} \leftarrow z$;
        $c_z \leftarrow c_z + 1$;
        $n_z^w \leftarrow n_z^w + 1$;
        $d_z \leftarrow d_z + w$;
        $n_{z,d} \leftarrow n_{z,d} + 1$;
foreach iteration do
    UpdateWordTopicProb() ;                                              /* See Eq. 9 and 11 */
    foreach $d \in D$ do
        UpdateTopMTopics() ;                                             /* See Eq. 5 */
    foreach $d \in D$ do
        foreach $w \in d$ do
            $z \leftarrow z_{d,w}$;
            $c_z \leftarrow c_z - 1$;
            $n_z^w \leftarrow n_z^w - 1$;
            $d_z \leftarrow d_z - \{w\}$;
            $n_{z,d} \leftarrow n_{z,d} - 1$;
            foreach Z do
                $\vec{\mathbb{Z}}_{d,w}(\mathbf{Z}) \leftarrow z \sim p(z_{d,w} = z|\vec{z}_{\neg(d,w)}, \mathbf{Z}, \vec{d})$;   /* See Eq. 3 */
        $\mathbf{Z}_d \leftarrow \mathbf{Z} \sim p(\mathbf{Z}_d = \mathbf{Z}|\vec{z}_{\neg d}, \vec{\mathbb{Z}}_d(\mathbf{Z}), \vec{d})$;   /* See Eq. 4 */
        foreach $w \in d$ do
            $z \leftarrow \vec{\mathbb{Z}}_{d,w}(\mathbf{Z}_d)$;
            $z_{d,w} \leftarrow z$;
            $c_z \leftarrow c_z + 1$;
            $n_z^w \leftarrow n_z^w + 1$;
            $d_z \leftarrow d_z + \{w\}$;
            $n_{z,d} \leftarrow n_{z,d} + 1$;
```

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

| Example word (Frequency) | Word (frequency, #co-occurrence with the 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) |

### 4.1. 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, word embeddings are learnt with the aim to retain words' global contextual information, hence the learnt word embeddings capture the general word co-occurrence patterns. That is, words that are semantically or syntactically similar to each other are projected to be closer in the latent space. Next, we illustrate this point using some example words.

Table II lists $5$ example words in the left-hand column (*i.e.,* “artworks”, “movie”, “fiction”, “company”, “stock”). The number that follows 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 5.1 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 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.

As 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.

### 4.2. Incorporating Word Embeddings by GPU

It has been shown 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 [Mimno et al. 2011]. 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 and GPU-PDMM achieve this through the generalized P$\acute{o}$lya urn model [Mahmoud 2008].

**Generalized P$\acute{o}$lya Urn Model** can be understood in terms of colored balls in a urn, where the probability of seeing a ball in each color is linearly proportional to the number of balls in that color in the urn. In a simple P$\acute{o}$lya urn model, when a ball in a particular color is sampled from the urn, the sampled ball along with a new ball in 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 P$\acute{o}$lya urn model, when a ball in a particular color is sampled, a certain number of balls in similar colors are put back along with the original ball and a new ball in that color. In this sense, the set of balls in 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 in 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 external thesauri or domain-specific knowledge for better topic inference in the standard LDA [Chen and Liu 2014; Chen et al. 2013a,b; Mimno et al. 2011]. However, these works rely on specific domain knowledge, which may restrict their usage in a broader range. Here, we exploit the combination of the 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 relation set $\mathbb{M}$ can be constructed, consisting of all word pairs whose semantic relatedness score is higher than a predefined threshold $\epsilon$, *i.e.,* $\mathbb{M} = \{\langle w_i, w_j \rangle | sr(w_i, w_j) > \epsilon\}$. The threshold $\epsilon$ is set to filter less semantic related word pairs.

As being a preliminary study on exploiting general semantic relation based on word embeddings, we fix the amount of promotion $\mu$ for each semantically related word $w'$ when working on word $w$ in this work. 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.

$$\mathbb{A}_{w,w'} = \begin{cases} 1 & w{=}w' \\ \mu & w' \in \mathbb{M}_w \text{ and } w' \neq w \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

The weight $\mu$ could be set based on the auxiliary word embeddings under use.

Incorporating the GPU model with DMM and PDMM gives us the proposed GPU-DMM and GPU-PDMM model respectively. 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. Although PDMM allows multiple topics to be associated with a short text, topic irrelevant words likely exist. 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:

$$\mathbb{S}_{d,w} \sim Bernoulli(\lambda_{w,z_d}) \tag{7}$$

$$\lambda_{w,z} = \frac{p(z|w)}{p_{max}(z'|w)} \tag{8}$$

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

$$p(z = k|w) = \frac{p(z = k)p(w|z = k)}{\sum_{i=1}^{K} p(z = i)p(w|z = i)} \tag{9}$$

In Equation 7, $\mathbb{S}_{d,w}$ indicates whether GPU is applied to word $w$ given document $d$ and topic $z_d$, where $\lambda_{w,z_d}$ is defined in Equation 8. Observe from Equation 8, $\mathbb{S}_{d,w}$ is strongly related to the ratio of the conditional topic probability $z_d$ given word $w$ to its 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 the GPU model. In the proposed word filtering strategy, calculating $p(z = k|w)$ (see Equation 9) for every word $w$ of a document and topic $k$ is

**ALGORITHM 2:** GPU-DMM

```
input  : Topic number $K$, $\alpha$, $\beta$, $\mu$, $\mathbb{M}$ and $D$ short documents
output: The posterior topic-word distribution
foreach $d \in D$ do
    $z_d \leftarrow z \sim Multinomial(1/K)$;
    $m_z \leftarrow m_z + 1$;
    $\tilde{n}_z \leftarrow \tilde{n}_z + 1$;
    foreach $w \in d$ do
        $\tilde{n}_z^w \leftarrow \tilde{n}_z^w + N_d^w$;
        $\mathbb{S}_{d,w} \leftarrow 0$;
foreach iteration do
    UpdateWordTopicProb() ;                                /* See Eq. 9 and 11 */
    foreach $d \in D$ do
        $z \leftarrow z_d$;
        $m_z \leftarrow m_z - 1$;
        foreach $w \in d$ do
            UpdateCounter ($\mathbb{S}_{d,w}$, $\mathbb{A}$, $d$, $w$, $False$);
        $z_d \leftarrow z \sim p(z_d = z|\vec{z}_{\neg d}, \vec{d})$;
        $m_z \leftarrow m_z + 1$;
        foreach $w \in d$ do
            $n_{z,d}^w \leftarrow N_d^w$;
            UpdateGPUFlag ($\mathbb{S}_{d,w}$);                        /* See Eq. 7 */
            UpdateCounter ($\mathbb{S}_{d,w}$, $\mathbb{A}$, $d$, $w$, $True$);
```

**ALGORITHM 3:** UpdateCounter ($\mathbb{S}_{d,w}, \mathbb{A}, d, w, promotion$)

```
if $promotion == True$ then
    if $\mathbb{S}_{d,w} == 1$ then                                  /* To apply GPU */
        foreach $w' \in \mathbb{M}_w$ do
            $\tilde{n}_z \leftarrow \tilde{n}_z + n_{z,d}^w \cdot \mathbb{A}_{w,w'}$;
            $\tilde{n}_z^{w'} \leftarrow \tilde{n}_z^{w'} + n_{z,d}^w \cdot \mathbb{A}_{w,w'}$;
    else
        $\tilde{n}_z \leftarrow \tilde{n}_z + 1$;
        $\tilde{n}_z^w \leftarrow \tilde{n}_z^w + 1$;
else
    if $\mathbb{S}_{d,w} == 1$ then                                /* GPU was applied */
        foreach $w' \in \mathbb{M}_w$ do
            $\tilde{n}_z \leftarrow \tilde{n}_z - n_{z,d}^w \cdot \mathbb{A}_{w,w'}$;
            $\tilde{n}_z^{w'} \leftarrow \tilde{n}_z^{w'} - n_{z,d}^w \cdot \mathbb{A}_{w,w'}$;
    else
        $\tilde{n}_z \leftarrow \tilde{n}_z - 1$;
        $\tilde{n}_z^w \leftarrow \tilde{n}_z^w - 1$;
```

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. Note that the word filtering applies to both GPU-DMM and GPU-PDMM.

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

The GPU model is nonexchangeable: 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.* [2011] [Mimno et al. 2011], 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:

$$p(z_d = k|\vec{z}_{\neg d}, \vec{d}) \propto \frac{m_{k,\neg d} + \alpha}{D - 1 + K\alpha} \times \frac{\prod_{w \in d} \prod_{j=1}^{N_d^w} (\tilde{n}_{k,\neg d}^w + \beta + j - 1)}{\prod_{i=1}^{N_d} (\tilde{n}_{k,\neg d} + V\beta + i - 1)} \tag{10}$$

In Equation 10, $\tilde{n}_{k,\neg d}$ 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 11:

$$p(w|z = k) = \frac{\tilde{n}_k^w + \beta}{\sum_w^V \tilde{n}_k^w + V\beta} \tag{11}$$

The details of the Gibbs sampling process of GPU-DMM are described in Algorithm 2. At first, GPU-DMM initializes the topic assignment for each document with a uniform distribution without applying the GPU model (Lines 1–7). 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 9 and 11 (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 10 (Lines 10-15). 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 7).

Shown in Algorithm 3, 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 3). Recall that the semantic relation set $\mathbb{M}_w$ of word $w$ also contains $w$ itself, *i.e.,* $w \in \mathbb{M}_w$ (see Equation 6). 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 3).

With the new sampled topic, the flag $\mathbb{S}_{d,w}$ for each word $w$ is updated based on Equation 7 (Line 19 in Algorithm 2). Then the corresponding counts for word $w$ are added based on the updated flag $\mathbb{S}_{d,w}$ through function *UpdateCounter()* (Line 20 in Algorithm 2). Similarly, if $\mathbb{S}_{d,w} = 1$, $w$ and all its related words are promoted. Otherwise, a simple update as in DMM is applied (Lines 2-8 in Algorithm 3). This iterative process continues until the predefined number of iterations is reached.

The Gibbs sampling process of GPU-PDMM, described in Algorithm 4, is similar to that of GPU-DMM with the following changes. The conditional distribution for Gibbs sampling in Equation 3 is rewritten for GPU-PDMM as follows:

$$p(z_{d,w} = k|\vec{z}_{\neg d,w}, \mathbf{Z}_d, \vec{d}) \propto \frac{1}{t_d} \times \frac{\tilde{n}^w_{k,\neg(d,w)} + \beta}{\sum_v^V \tilde{n}^v_{k,\neg(d,w)} + V\beta} \tag{12}$$

The conditional distribution on $\mathbf{Z}_d$ in Equation 4 is rewritten as follows:

$$p(\mathbf{Z}_d|\vec{z}_{\neg d}, \vec{z}_d, \vec{d}) \propto$$

$$\frac{\lambda^{t_d}}{t_d^{n_d}} \times \frac{\prod_{k\in\mathbf{Z}_d}(c_{k,\neg d} + \alpha)}{\prod_{i=0}^{t_d-1}(\sum_k^K c_{k,\neg d} + K\alpha - i)} \times \prod_{k\in\mathbf{Z}_d} \frac{\prod_{w\in d_k}\prod_{i=0}^{n^w_{k,d}}(\tilde{n}^w_{k,\neg d} + n^w_{k,d}) - i + \beta}{\prod_{i=0}^{n_{k,d}-1}(\sum_w^V \tilde{n}^w_{k,\neg d} + n_{k,d} - i + V\beta)} \tag{13}$$

**Model Complexity.** We now analyze the time complexity of PDMM, GPU-DMM and GPU-PDMM 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. Without pruning strategy, PDMM needs to sample each $z_{d,w}$ for all possible $\mathbf{Z}_d$ settings, resulting in a time complexity of $O(D\bar{\ell}\sum_{i=1}^{\tau-1} C_K^i)$[3]. With pruning strategy, the time complexity of PDMM becomes $O(D\bar{\ell}\sum_{i=1}^{\tau-1} C_M^i)$ in an iteration. Note that $M$ is typically much smaller than $K$.

Extended with GPU model, GPU-DMM has a time complexity of $O(KD\bar{\ell} + D\bar{\ell}\tau + KV)$, where $\tau$ is a coefficient by considering the cost involved by applying the GPU model. Similarly, the time complexity of GPU-PDMM without pruning strategy is $O(D\bar{\ell}\sum_{i=1}^{\tau-1} C_K^i + D\bar{\ell}\tau + KV)$, and with pruning strategy it is $O(D\bar{\ell}\sum_{i=1}^{\tau-1} C_M^i + D\bar{\ell}\tau + KV)$. The value of $\tau$ depends on: (i) the average number of words for which the GPU model is applied based on Equation 7, 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 relation set $\mathbb{M}_w$ 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, $\tau$ is expected to be relatively small. Furthermore, it is expected that $KV < KD\bar{\ell}$. Hence, GPU-DMM does not add too much computational cost to DMM, and the same applies to GPU-PDMM compared with PDMM.

## 5. EXPERIMENT

In this section, we conduct experiments to evaluate the proposed PDMM, GPU-DMM, and GPU-PDMM, against the state-of-the-art alternatives.[4] The performance in terms of topic coherence and document classification are reported over two publicly available datasets, *i.e.,* a Web search snippet dataset in English and a Q&A dataset in Chinese. We also report the time taken per iteration for all models evaluated in our experiments. The experimental results show that our proposed models achieve promising performance in both effectiveness and efficiency.

[3] $C_k^i$ refers to the number of $i$-combinations from $k$ elements.

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

**ALGORITHM 4:** GPU-PDMM

```
input  : Topic number $K$, $\alpha$, $\beta$, $\lambda$, $M$, $\mathbb{M}$ and $D$ short documents
output: The posterior topic-word distribution
foreach $d \in D$ do
    $t_d \leftarrow t \sim Poisson(\lambda)$;
    sample $t_d$ distinct topics $\mathbf{Z}_d \sim Multinomial(1/K)$;
    foreach $w \in d$ do
        uniformly sample $z \sim \mathbf{Z}_d$;
        $z_{d,w} \leftarrow z$;
        $c_z \leftarrow c_z + 1$;
        $n_z^w \leftarrow n_z^w + 1$;
        $d_z \leftarrow d_z + w$;
        $n_{z,d} \leftarrow n_{z,d} + 1$;
        $n_{z,d}^w \leftarrow n_{z,d}^w + 1$;
        $\mathbb{S}_{d,w} \leftarrow 0$;
foreach iteration do
    UpdateWordTopicProb() ;                                    /* See Eq. 9 and 11 */
    foreach $d \in D$ do
        UpdateTopMTopics() ;                                   /* See Eq. 5 */
    foreach $d \in D$ do
        foreach $w \in d$ do
            $z \leftarrow z_{d,w}$;
            $c_z \leftarrow c_z - 1$;
            $n_z^w \leftarrow n_z^w - 1$;
            $d_z \leftarrow d_z - \{w\}$;
            $n_{z,d} \leftarrow n_{z,d} - 1$;
            UpdateCounter ($\mathbb{S}_{d,w}$, $\mathbb{A}$, $d$, $w$, $False$);
            $n_{z,d}^w \leftarrow n_{z,d}^w - 1$;
            foreach Z do
                $\vec{\mathbb{Z}}_{d,w}(\mathbf{Z}) \leftarrow z \sim p(z_{d,w} = z|\vec{z}_{\neg(d,w)}, \mathbf{Z}, \vec{d})$;   /* See Eq. 3 */
        $\mathbf{Z}_d \leftarrow \mathbf{Z} \sim p(\mathbf{Z}_d = \mathbf{Z}|\vec{z}_{\neg d}, \vec{\mathbb{Z}}_d(\mathbf{Z}), \vec{d})$;   /* See Eq. 4 */
        foreach $w \in d$ do
            $z \leftarrow \vec{\mathbb{Z}}_{d,w}(\mathbf{Z}_d)$;
            $z_{d,w} \leftarrow z$;
            $c_z \leftarrow c_z + 1$;
            $n_z^w \leftarrow n_z^w + 1$;
            $d_z \leftarrow d_z + \{w\}$;
            $n_{z,d} \leftarrow n_{z,d} + 1$;
            $n_{z,d}^w \leftarrow n_{z,d}^w + 1$;
            UpdateGPUFlag ($\mathbb{S}_{d,w}$);                  /* See Eq. 7 */
            UpdateCounter ($\mathbb{S}_{d,w}$, $\mathbb{A}$, $d$, $w$, $True$);
```

### 5.1. Datasets

**BaiduQA** dataset is a collection of $648,514$ questions crawled from a popular Chinese Q&A website[5]. Each question is annotated with a category label by its asker. This dataset was prepared and has been previously used in [Cheng et al. 2014; Yan et al. 2013]. The dataset was preprocessed by the authors, *e.g.,* Chinese word segmentation

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

Table III. Statistics on the two datasets. #Labels/categories: the number of ground truth labels; #Documents: the total number of documents; #Words per document: average length of documents in number of words.

| Dataset | #Labels/Categories | #Documents | #Words per document | Vocabulary |
|---|---|---|---|---|
| Snippet | 8 | 12,265 | 10.72 | 5,581 |
| BaiduQA | 35 | 179,042 | 4.11 | 26,560 |

was applied and duplicate words were removed. 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 [Chen et al. 2011; Phan et al. 2008; Sun 2012]. 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[6]; (3) remove 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 III. Observe that the BaiduQA dataset is much sparser, with a larger vocabulary, and its documents are much shorter compared to the Snippet dataset.

### 5.2. Experimental Setup

**Word Embeddings.** For the Snippet dataset, we use the pre-trained $300$-dimensional word embeddings from the Google News corpus[7]. For the BaiduQA dataset, we train 100-dimensional word embeddings from 7 million Chinese articles crawled from Baike website[8], using Google's Word2Vec toolkit with Skip-gram algorithm [Mikolov et al. 2013]. If a word has no embedding, the word is considered having no semantically related words. The BaiduQA dataset was preprocessed with Chinese word segmentation by the dataset provider [Cheng et al. 2014; Yan et al. 2013]. For learning word embeddings and evaluating topic coherence, we used ICTCLAS[9], 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.

As mentioned in Section 4.1, a parameter $\epsilon$ determines the semantic relations provided by the auxiliary word embeddings. We employ a simple process to set the value of $\epsilon$, 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 $\epsilon$ value after examining these related words. The related words with similarity score larger than this value hold strong semantic or syntactic relation to the selected word. On the other hand, the words with similarity score below this value are considered less related to the selected words. Based on this manual examination process, we set $\epsilon = 0.5$ on the Snippet dataset, and $\epsilon = 0.7$ on the BaiduQA datasets, for both models GPU-DMM and GPU-PDMM. We argue that the optimal value of $\epsilon$ depends on the external text corpus, the language, and the algo-

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

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

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

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

rithm used in learning the word embeddings. Therefore, the aforementioned manual examination process for $\epsilon$ selection is necessary.

After setting $\epsilon$, we observe that for some words, the number of semantically related words (*i.e.,* the size of $\mathbb{M}_w$) is very large. For example, there are $129$ words semantically related to the word *houston*, including *dallas*, *atlanta*, *orlando*, and *cleveland*. In fact, most of these $129$ words refer to geographic areas like the word *houston* itself. Another example is the word *carl*, which is highly related to many English names like *gordon*, *henry*, and*james*. Such words often refer to semantic relations in some specific domains (*e.g.,* geographical, temporal, or person names) rather than general concepts as the examples listed in Table II. Hence, in our experiments, we simply disregard all the related words in $\mathbb{M}_w$ if $|\mathbb{M}_w| > 20$ [10].

As shown in the model overview in Figure 1, the preparation of word embeddings and $\mathbb{M}$ is independent from the proposed GPU-DMM and GPU-PDMM models. That is, the preparation is offline, and does not contribute to the computational cost of the models.

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

- **Biterm Topic Model (BTM)** learns topics by directly modeling the generation of word co-occurrence patterns in the short text corpus [Yan et al. 2013]. In BTM, a *biterm* is an unordered word pair that co-occurs 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 [Quan et al. 2015]. It requires one 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 5.4) and set it to $200$ and $700$ respectively, on the Snippet and BaiduQA datasets, for achieving the best classification results.
- **Pseudo-document-based Topic Model (PTM)** is a recent self-aggregation method for short text topic modeling [Zuo et al. 2016a]. We use the recommended setting by the authors and set the number of pseudo-documents to $1000$.
- **Word Network Topic Model (WNTM)** infers the latent topics of short texts explicitly based on the word co-occurrence network [Zuo et al. 2016b].
- **Dirichlet Multinomial Mixture (DMM)** assumes that each short document has only one topic, and works as the basis of our model [Yin and Wang 2014].
- **Gaussian Mixture Topic Model (GMTM)** utilizes a Gaussian mixture model (GMM) to learn a parametric model for the words in terms of word embeddings [Sridhar 2015]. Each component of the GMM is equivalent to a latent topic.
- **Latent Concept Topic Model (LCTM)** learns the latent topics of short texts with a latent concept layer by using word embeddings [Hu and Tsujii 2016]. The optimal parameter settings recommended by the authors are used here.
- **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 [Nguyen et al. 2015]. A hard constraint in LF-DMM is that all words in the short text corpus should have their corresponding word embeddings. Therefore we remove all the words without word embeddings. We use the implementation provid-

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

ed by the authors and use the recommended settings with $\lambda = 0.6$, $\alpha = 0.1$, $\beta = 0.01$ as in their paper[11].

As for GPU-DMM and GPU-PDMM, we need to set the amount of promotion $\mu$ for each semantically related word. We empirically set $\mu$ to be $0.1$ and $0.3$, respectively on the BaiduQA and Snippet datasets, for both models. Unless explicitly specified, $\tau = 2$, $\lambda = 1.5$ and $M = 10$ are used for PDMM, and GPU-PDMM.

Note that Das *et al.* [2015] also propose an 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 [Hong and Davison 2010; Quan et al. 2015; Zhao et al. 2011]. Hence, we exclude this model from the comparison.

We run Gibbs sampling for $1,000$ iterations in each run, and report the average results over $5$ runs, for all methods. The only exception is $2,000$ iterations ($1,500$ iterations with baseline model + $500$ iterations with LF-DMM) for LF-DMM, as in its original paper [Nguyen et al. 2015]. 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*.

### 5.3. Evaluation in terms of 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 [Chang et al. 2009], 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 [Mimno et al. 2011].

Following Cheng *et al.* [2014], we use the PMI-Score proposed in [Newman et al. 2010] 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:

$$PMI(k) = \frac{2}{T(T-1)} \sum_{1 \le i < j \le T} \log \frac{p(w_i, w_j)}{p(w_i)p(w_j)} \qquad (14)$$

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 better learnt topics. Note that an external corpus is needed to calculate the PMI-Scores in Equation 14. In our experiments, we use 3 million English Wikipedia articles and 7 million Chinese Baike articles for the Snippet and BaiduQA datasets, respectively.

Table IV reports 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.[12] Here, we make several observations.

First, on both datasets, GPU-PDMM achieves the best topic coherence across all settings, and the improvement over other models are statistical significance at $0.05$ level, except on three cases (*i.e.,* LF-DMM at $K = 40, T = 5$ and $K = 80, T = 5$, and GPU-DMM at $K = 40, T = 10$). The performance gain by GPU-PDMM is in the range of $6.2\%$–$96.7\%$ and $3.8\%$–$101.8\%$ on the Snippet and BaiduQA datasets respectively. GPU-DMM is the second best model in most cases, and always outperforms GMTM, SATM and DMM. Specifically, GPU-DMM outperforms all baselines across all settings

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

[12] Because LCTM does not directly infer topic-word distribution, we exclude it from this comparison.

Table IV. Topic coherence 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. † and $*$ indicates that the difference to the best result is statistically significant at $0.01$ and $0.05$ level respectively.

| Dataset | Model | $K$=40 | | | $K$=60 | | | $K$=80 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $T$ = 5 | $T$ = 10 | $T$ = 20 | $T$ = 5 | $T$ = 10 | $T$ = 20 | $T$ = 5 | $T$ = 10 | $T$ = 20 |
| Snippet | BTM | $1.7602^\dagger$ | $1.7086^\dagger$ | $1.6262^\dagger$ | $1.8076^\dagger$ | $1.7318^\dagger$ | $1.6616^\dagger$ | $1.8618^\dagger$ | $1.7292^\dagger$ | $1.6284^\dagger$ |
| | PTM | $1.6405^\dagger$ | $1.6115^\dagger$ | $1.6190^\dagger$ | $1.6932^\dagger$ | $1.6621^\dagger$ | $1.6443^\dagger$ | $1.6871^\dagger$ | $1.6624^\dagger$ | $1.6567^\dagger$ |
| | WNTM | $1.7971^\dagger$ | $1.7404^\dagger$ | $1.6836^\dagger$ | $1.8746^\dagger$ | $1.7684^\dagger$ | $1.7177^\dagger$ | $1.8953^\dagger$ | $1.7910^\dagger$ | $1.7093^\dagger$ |
| | GMTM | $1.2206^\dagger$ | $1.1598^\dagger$ | $1.1584^\dagger$ | $1.2948^\dagger$ | $1.2101^\dagger$ | $1.1991^\dagger$ | $1.3285^\dagger$ | $1.2198^\dagger$ | $1.2114^\dagger$ |
| | SATM | $1.9480^\dagger$ | $1.7680^\dagger$ | $1.6120^\dagger$ | $1.7146^\dagger$ | $1.5982^\dagger$ | $1.5312^\dagger$ | $1.6338^\dagger$ | $1.5432^\dagger$ | $1.4816^\dagger$ |
| | LF-DMM | $1.6798^\dagger$ | $1.6304^\dagger$ | $1.5676^\dagger$ | $1.6942^\dagger$ | $1.6166^\dagger$ | $1.5570^\dagger$ | $1.7280^\dagger$ | $1.6306^\dagger$ | $1.5606^\dagger$ |
| | DMM | $1.6936^\dagger$ | $1.6062^\dagger$ | $1.5312^\dagger$ | $1.7822^\dagger$ | $1.6408^\dagger$ | $1.5656^\dagger$ | $1.8406^\dagger$ | $1.6882^\dagger$ | $1.5380^\dagger$ |
| | PDMM | $1.8449^\dagger$ | $1.7383^\dagger$ | $1.7286^\dagger$ | $1.9565^\dagger$ | $1.8659^\dagger$ | $1.7968^\dagger$ | $2.0010^\dagger$ | $1.8989^\dagger$ | $1.7946^\dagger$ |
| | GPU-DMM | $\underline{2.0602}^\dagger$ | $\underline{2.0466}^\dagger$ | $\underline{1.9844}^\dagger$ | $\underline{2.1536}^\dagger$ | $\underline{2.1360}^\dagger$ | $\underline{2.0158}^\dagger$ | $\underline{2.1842}^\dagger$ | $\underline{2.1304}^\dagger$ | $\underline{2.0152}^\dagger$ |
| | GPU-PDMM | **2.2126** | **2.2813** | **2.1919** | **2.3260** | **2.3241** | **2.2147** | **2.3185** | **2.2998** | **2.1635** |
| BaiduQA | BTM | $1.7640^*$ | $1.6246^\dagger$ | $1.5294^\dagger$ | $1.6646^\dagger$ | $1.6058^\dagger$ | $1.5260^\dagger$ | $1.6990^*$ | $1.6220^\dagger$ | $1.5320^\dagger$ |
| | PTM | $1.6507^\dagger$ | $1.5646^\dagger$ | $1.4497^\dagger$ | $1.6278^\dagger$ | $1.5243^\dagger$ | $1.4616^\dagger$ | $1.6072^\dagger$ | $1.5207^\dagger$ | 1.4974 |
| | WNTM | $1.7029^\dagger$ | $1.5789^\dagger$ | $1.5055^\dagger$ | $1.6281^\dagger$ | $1.5412^\dagger$ | $1.5111^\dagger$ | $1.7081^*$ | $1.5978^\dagger$ | $1.5114^\dagger$ |
| | GMTM | $1.3974^\dagger$ | $1.3413^\dagger$ | $1.2929^\dagger$ | $1.2965^\dagger$ | $1.3073^\dagger$ | $1.2222^\dagger$ | $1.3691^\dagger$ | $1.3107^\dagger$ | $1.2496^\dagger$ |
| | SATM | $1.5360^\dagger$ | $1.3800^\dagger$ | $1.2540^\dagger$ | $1.3440^\dagger$ | $1.1000^\dagger$ | $1.0070^\dagger$ | $1.2380^\dagger$ | $1.1600^\dagger$ | $0.9380^\dagger$ |
| | LF-DMM | $\underline{1.7868}$ | $1.6804^*$ | $1.5834^\dagger$ | $\underline{1.7704}^\dagger$ | $\underline{1.6564}^\dagger$ | $1.5820^\dagger$ | $\underline{1.7526}$ | $\underline{1.6748}^\dagger$ | $\underline{1.5934}^\dagger$ |
| | DMM | $1.6014^\dagger$ | $1.4642^\dagger$ | $1.3304^\dagger$ | $1.4564^\dagger$ | $1.2312^\dagger$ | $1.2176^\dagger$ | $1.3974^\dagger$ | $1.0310^\dagger$ | $1.0586^\dagger$ |
| | PDMM | $1.7338^*$ | $1.6086^\dagger$ | $1.5306^\dagger$ | $1.6822^\dagger$ | $1.5350^\dagger$ | $1.4922^\dagger$ | $1.6340^*$ | $1.3764^\dagger$ | $1.3305^\dagger$ |
| | GPU-DMM | $1.7500^*$ | $\underline{1.7070}$ | $\underline{1.7244}^*$ | $1.7280^\dagger$ | $1.6228^\dagger$ | $\underline{1.6408}^\dagger$ | $1.5394^\dagger$ | $1.4992^\dagger$ | $1.5140^\dagger$ |
| | GPU-PDMM | **1.8545** | **1.7743** | **1.8214** | **1.8710** | **1.8786** | **1.9465** | **1.8357** | **1.8192** | **1.8931** |

on the Snippet dataset. On the BaiduQA dataset, LF-DMM is the second best in $6$ out of $9$ settings. GPU-DMM achieves the second best performance in the remaining three settings. This validates that incorporating general word semantic relations is beneficial for topic modeling. We also observe that GMTM delivers almost the worst performance across all settings on two datasets. GMTM learns the word embeddings over the target document collection itself. Since words with high semantic relatedness may not frequently co-occur in short texts, the word embeddings learnt from the same text collection could not provide additional semantic knowledge.

Second, significant performance gains are achieved by PDMM over DMM, across all the settings on the two datasets at $0.01$ level. Specifically, the performance gain by PDMM over DMM is about $8.2\%$–$16.7\%$ and $8.3\%$–$33.5\%$ on the Snippet and BaiduQA datasets respectively. The superiority of PDMM suggests that associating multiple topics with a short text helps uncover the underlying semantics in short texts.

Third, GPU-DMM and GPU-PDMM significantly outperform DMM and PDMM respectively on the two datasets. The improvement by GPU-DMM and GPU-PDMM is around $18.7\%$–$31.0\%$/$9.3\%$–$45.4\%$ and $15.9\%$–$31.2\%$/$7.0\%$–$42.3\%$ on the Snippet and BaiduQA respectively. This illustrates that the GPU mechanism is an appropriate s-trategy to exploit word semantic relations offered by word embeddings.

### 5.4. Evaluation in terms of 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 topic-level representation, as an indirect evaluation. A better classification accuracy means the learnt topics are more discriminative and representative. Here, we employ a linear kernel Support Vector Machines (SVM) classifier in sklearn[13] with default parameter settings. The classification accuracy is computed through $5$-fold cross validation on both datasets.

**Methods for Topic-level Representation.** DMM model and its variants 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. On the other hand, 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 [Quan et al. 2015]. There are two ways to infer $p(z|d)$:

— Naïve Bayes (NB) rule:

$$p(z=k|d) \propto p(z=k)\prod_{i=1}^{N_d} p(w_i|z=k) \tag{15}$$

— Summation over words (SW): The same as Equation 5.

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

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* $b$ [Yan et al. 2013]. However, in their comparison, they use the NB method for DMM instead. In [Quan et al. 2015], SW method is used for DMM and LDA and proven to largely improve the classification accuracy. However, there is no thorough comparison of the two methods for inferring $p(z|d)$.

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

Table V. Classification accuracy of two post inference methods (NB and SW) on the 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 VI. 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 |

Next, we first compare the two document representation methods by using DMM, GPU-DMM, and BTM on the Snippet dataset. Table V reports the classification accuracy by 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 V also shows that GPU-DMM achieves the best classification accuracy over all settings, followed by DMM. Similar observations hold on the BaiduQA dataset (results not shown).

To better understand the big difference in classification accuracy by using the two different representations NB and SW, we show the top $3$ topics with the corresponding $p(z|d)$ values for a random document in the Snippet dataset, by the two methods in Table VI. 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. 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 $1$ [Niculescu-Mizil and Caruana 2005]. 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 VII reports the document classification accuracy on the two datasets by using the $11$ models in comparison. We make the following observations.

First, On the Snippet dataset, our GPU-PDMM model significantly outperforms existing state-of-the-art models, followed by GPU-DMM. Also, PDMM and GPU-PDMM perform the second best on the BaiduQA dataset. Note that both GPU-DMM and GPU-PDMM outperform DMM and PDMM respectively on both datasets. GMTM, LCTM and LF-DMM all incorporate word embeddings in their models in one way or another. However, both GPU-DMM and GPU-PDMM outperform these models significantly. These observations validate that incorporating general word semantic relations with the GPU model is beneficial for short text topic modeling.

Second, in our experiments, DMM is the best performing model in the document classification task among the existing state-of-the-art alternatives on the Snippet

Table VII. Average classification accuracy of each model on two datasets, with different number of topic $K$ settings. The best and second best results are highlighted in boldface and underlined respectively on each dataset. † indicates that the difference with the best result is statistically significant at $0.01$ level.

| Model | Snippet | | | BaiduQA | | |
|---|---|---|---|---|---|---|
| | $K$ = 40 | $K$ = 60 | $K$ = 80 | $K$ = 40 | $K$ = 60 | $K$ = 80 |
| BTM | 0.8100$^\dagger$ | 0.8183$^\dagger$ | 0.8272$^\dagger$ | 0.5098$^\dagger$ | 0.5336$^\dagger$ | 0.5486$^\dagger$ |
| PTM | 0.8191$^\dagger$ | 0.8312$^\dagger$ | 0.8322$^\dagger$ | 0.5228$^\dagger$ | 0.5542$^\dagger$ | 0.5725$^\dagger$ |
| WNTM | 0.8261$^\dagger$ | 0.8296$^\dagger$ | 0.8259$^\dagger$ | **0.5645** | **0.5926** | **0.6027** |
| GMTM | 0.8192$^\dagger$ | 0.8266$^\dagger$ | 0.8275$^\dagger$ | 0.4446$^\dagger$ | 0.4807$^\dagger$ | 0.5061$^\dagger$ |
| LCTM | 0.7529$^\dagger$ | 0.7880$^\dagger$ | 0.7862$^\dagger$ | 0.2321$^\dagger$ | 0.2573$^\dagger$ | 0.2630$^\dagger$ |
| SATM | 0.8284$^\dagger$ | 0.8231$^\dagger$ | 0.8235$^\dagger$ | 0.4872$^\dagger$ | 0.4567$^\dagger$ | 0.4605$^\dagger$ |
| LF-DMM | 0.7414$^\dagger$ | 0.7409$^\dagger$ | 0.7462$^\dagger$ | 0.4240$^\dagger$ | 0.4486$^\dagger$ | 0.4868$^\dagger$ |
| DMM | 0.8298$^\dagger$ | 0.8522$^\dagger$ | 0.8479$^\dagger$ | 0.5228$^\dagger$ | 0.5476$^\dagger$ | 0.5532$^\dagger$ |
| PDMM | 0.8521 | 0.8563$^\dagger$ | 0.8676$^\dagger$ | 0.5528$^\dagger$ | 0.5624$^\dagger$ | 0.5767$^\dagger$ |
| GPU-DMM | 0.8539 | 0.8667 | 0.8722 | 0.5439$^\dagger$ | 0.5637$^\dagger$ | 0.5708$^\dagger$ |
| GPU-PDMM | **0.8608** | **0.8683** | **0.8744** | 0.5445$^\dagger$ | 0.5685$^\dagger$ | 0.5833$^\dagger$ |

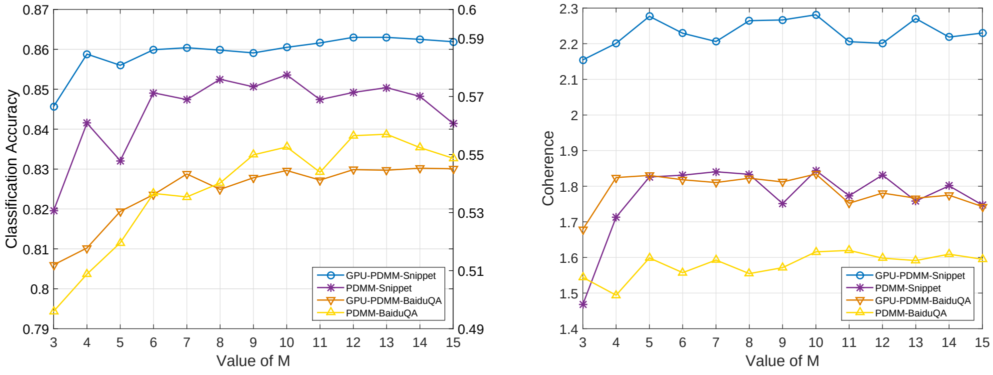
dataset. It performs the third best on the BaiduQA dataset as well. Although BTM was shown to perform better than DMM in [Yan et al. 2013], as we discussed earlier, the NB based document representation was used for DMM in [Yan et al. 2013], which may be the reason for the poorer performance. By using 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 in little discriminative word co-occurrence knowledge, which is also observed in [Quan et al. 2015; Yunqing et al. 2015].

Third, WNTM achieves the best classification accuracy on the BaiduQA dataset. Note that, on average there are only 4.11 words per document on the BaiduQA (see Table III). This suggests that exploiting the word co-occurrence network is beneficial for extremely short documents (*e.g.,* documents in BaiduQA). SATM and GMTM deliver mixed performance on the two datasets. On the Snippet dataset, they performs slightly better than BTM, but perform much poorer than BTM on the BaiduQA dataset. Surprisingly, we observe that both LF-DMM and LCTM perform the worst among all models on both datasets even though they also exploit word embeddings. As to LF-DMM, 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. As to LCTM, since the documents in the BaiduQA are much shorter and therefore sparser, the addition of latent concept layer would exacerbate the sparsity problem, leading to much worse topic inference.

Compared to our previous work [Li et al. 2016], a new observation is that both GPU-PDMM and PDMM outperform GPU-DMM and DMM respectively across the two datasets. Similar observation has been made in Section 5.3. This validates again that allowing more than one topic within a short document is beneficial for short text topic modeling.

### 5.5. Parameter Analysis

We now study the impact of the parameter settings (*i.e.,* $M, \mu, \tau, \lambda$) in PDMM and GPU-PDMM, in terms of classification accuracy and topic coherence. All experiments are conducted by using number of topics $K = 40$ and number of top words per topic $T = 10$. Similar performance patterns are also observed with other $K$ values. In the next section we will report on a comparison in terms of efficiency.



(a) Classification Accuracy (b) Topic Coherence

Fig. 3. Effect of parameter $M$ with the setting of $K = 40$ and $T = 10$.

**Impact of $M$ value.** The $M$ value controls the size of the search space of $\mathbf{Z}_d$ for PDMM and GPU-PDMM. A small $M$ value leads to better efficiency, but results in a sacrifice of topic quality since the optimal topics may be filtered out. On the other hand, an unnecessarily large $M$ value results in high computation cost and incurs more noise. Figure 3 depicts the classification accuracy and topic coherence of PDMM and GPU-PDMM by varying $M$ value from $3$ to $15$. As to the classification accuracy, shown in Figure 3(a), both PDMM and GPU-PDMM deliver better classification performance as $M$ increases in the range of $[3, 10]$. Both models reach stable performance when $M$ is in the range of $[8, 13]$. However, when $M$ is much larger (*i.e.,* $M > 13$), the two models experience performance degradation. This is reasonable because larger $M$ introduces more irrelevant topics and adversely impacts the topic inference process. A similar pattern is also observed for topic coherence, as shown in Figure 3(b). Both models reach stable topic coherence scores when $M \geq 5$. However, it is observed that performance decreases when $M > 11$. Based on the results, we set $M = 10$ in our experiments.

**Impact of $\mu$ value.** We now study the effect of the weight $\mu$ in GPU-PDMM (see Equation 6). For better understanding of the impact of $\mu$, we also include the performance of GPU-DMM reported in [Li et al. 2016] for a comprehensive comparison. Figure 4 plots the classification accuracy of setting different $\mu$ values under $K = 40$ on the two datasets, with topics learned by the two models. Note that when $\mu = 0$, GPU-DMM and GPU-PDMM are equivalent to the DMM and PDMM models respectively.

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

**Impact of the $\tau$ value.** The $\tau$ value specifies the maximum number of topics a short document can have (*i.e.,* $t_d \leq \tau$). Note that PDMM and GPU-PDMM degrade to DMM

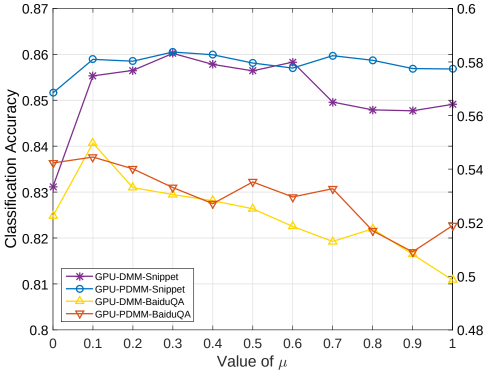


Fig. 4. Effect of weight $\mu$ under $K = 40$

Table VIII. Classification accuracy of PDMM and GPU-PDMM on two datasets, with varying $\tau$ values. The best results are highlighted in boldface on each dataset.

| Dataset | Model | $\tau$ = 1 | $\tau$ = 2 | $\tau$ = 3 | $\tau$ = 4 |
|---|---|---|---|---|---|
| Snippet | PDMM | 0.8298 | **0.8521** | 0.8463 | 0.8461 |
| | GPU-PDMM | 0.8539 | 0.8608 | **0.8624** | 0.8506 |
| BaiduQA | PDMM | 0.5228 | **0.5528** | 0.5373 | 0.5447 |
| | GPU-PDMM | 0.5445 | **0.5445** | 0.5358 | 0.5361 |

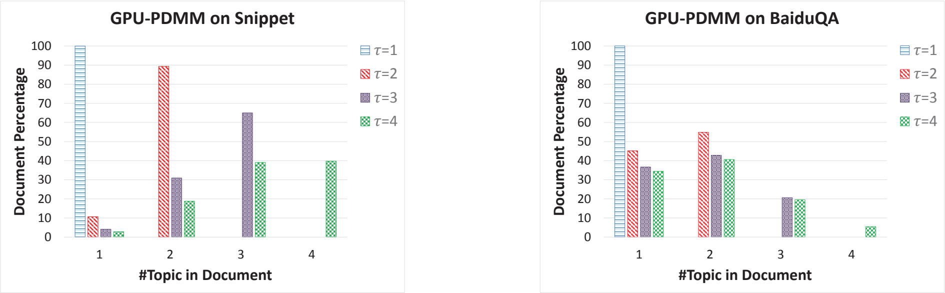
and GPU-DMM respectively when $\tau = 1$. On the other hand, when $\tau \to K$, the models are similar to the standard LDA model with a constrain on the global document-topic distribution. Tables VIII and IX report the classification accuracy and topic coherence of PDMM and GPU-PDMM by varying $\tau$ value from $1$ to $4$. From Table VIII, we observe that the optimal classification performance is achieved by setting $\tau = 2$. A larger $\tau$ value results in significant performance degradation since the search space of $\mathbf{Z}_d$ becomes larger and more irrelevant topics could be sampled. On the other hand, we observe that both PDMM and GPU-PDMM deliver stable coherence scores at $\tau = 2, 3, 4$ in Table IX. This suggests that allowing multiple topics within a short text helps to uncover the coherent topics.

Since PDMM and GPU-PDMM allow each document to be generated by more than a single topic, it is interesting to investigate the distributions of topic numbers that are discovered for each short document by the two models. Figure 5 illustrates the topic number distribution by using GPU-PDMM on the Snippet and BaiduQA datasets respectively. On the Snippet dataset, about $10\%$ of the documents are associated with a single topic when $\tau = 2$. The number decreases to $4.3\%$ when $\tau = 4$, *i.e.,* most documents are associated with more than one topic. On the BaiduQA dataset, even with $\tau = 4$, about $75\%$ of documents are associated with one or two topics only. A similar pattern is also observed for PDMM (results not shown here). Note that the documents in the Snippet dataset are much longer than documents in the BaiduQA dataset. It is reasonable that documents in the Snippet could cover more topics than the documents in the BaiduQA. Overall, the results suggest that the proposed PDMM and GPU-PDMM can successfully identify the correct topics within each short document. Based on the results, we set $\tau = 2$ in our experiments.

**Impact of the $\lambda$ value.** The parameter $\lambda$ of the Poisson distribution in PDMM and GPU-PDMM specifies the prior average number of topics a short document could be associated with. Figure 6 plots the classification accuracy and topic coherence of PDM-

Table IX. Topic coherence ($T = 10$) of PDMM and GPU-PDMM on two datasets, with varying $\tau$ values. The best results are highlighted in boldface on each dataset.

| Dataset | Model | $\tau$ = 1 | $\tau$ = 2 | $\tau$ = 3 | $\tau$ = 4 |
|---|---|---|---|---|---|
| Snippet | PDMM | 1.6004 | 1.7383 | 1.8513 | **1.8851** |
| | GPU-PDMM | 2.1388 | 2.2813 | 2.2866 | **2.3498** |
| BaiduQA | PDMM | 1.5052 | **1.6086** | 1.6079 | 1.5831 |
| | GPU-PDMM | 1.7413 | 1.7743 | **1.8363** | 1.7713 |



(a) Topic Number Distribution on Snippet

(b) Topic Number Distribution on BaiduQA

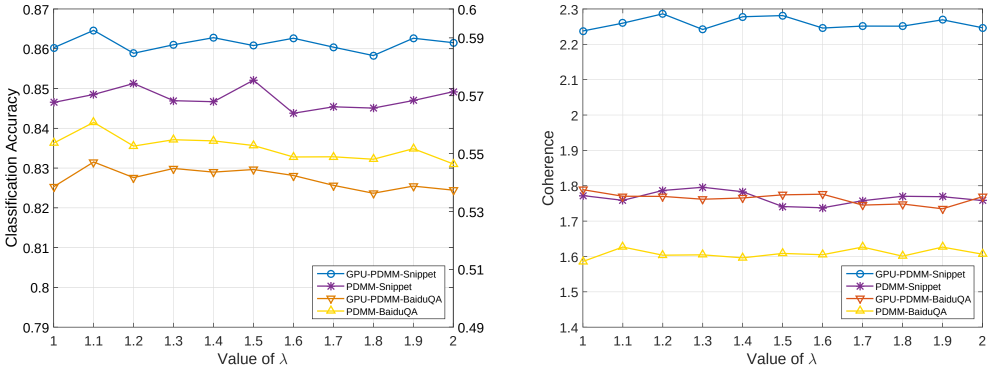
Fig. 5. Topic Number Distribution of GPU-PDMM

M and GPU-PDMM by varying $\lambda$ value from $1.0$ to $2.0$. As to the classification accuracy, shown in Figure 6(a), both PDMM and GPU-PDMM achieve very close performance in the range. The performance degrades slightly when larger $\lambda$ values (*i.e.,* $\lambda > 1.5$) are used on the BaiduQA dataset. Because the documents in the BaiduQA dataset are much shorter, it is reasonable that a prior indicating more topics per document may not perform well. Note that the performance degradation is much milder for GPU-PDMM than PDMM. Similar observation holds on the Snippet dataset. This suggests that, by incorporating general word semantic relations, GPU-PDMM is more robust against inappropriate parameter settings. As to the topic coherence, we can see in Figure 6(b) that both PDMM and GPU-PDMM achieve a stable performance on both datasets. Based on the results, we set $\lambda = 1.5$ in our experiments.

### 5.6. Efficiency

In this set of experiments, we compare the running time of all models. We implement PDMM, GPU-DMM, GPU-PDMM, BTM, PTM, WNTM, GMTM, LCTM, SATM and DMM models in Java, and use the Java implementation of LF-DMM provided by its authors.

Table X reports the average run-time (in seconds) per iteration for each model. We first discuss the run-time of the three models DMM, GPU-DMM, and GPU-PDMM. The simplest model, DMM, is the most efficient one as expected. GPU-DMM is slightly slower than DMM, being the second most efficient model on the Snippet dataset. Its run-time is very close to the second best model (*i.e.,* WNTM) on the BaiduQA dataset. 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 at most three times that of DMM for different topic numbers $K$ on both datasets. PDMM and GPU-PDMM take much more time for one iteration than



(a) Classification Accuracy (b) Topic Coherence

Fig. 6. Effect of parameter $\lambda$ with the setting of $K = 40$ and $T = 10$.

DMM and GPU-DMM respectively. In particular, GPU-PDMM is the slowest model on the BaiduQA dataset among the three models. This observation is consistent with the analysis made in Section 4.2, because PDMM and GPU-PDMM has a much larger search space of $\mathbf{Z}_d$.

Among all models evaluated, LCTM is the slowest model on both datasets. This is expected since it involves the probability calculation of multivariate Gaussian distributions. SATM is much faster than BTM on the Snippet dataset, but much slower than BTM on the BaiduQA dataset. As described in [Quan et al. 2015], 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. A similar observation can be made for WNTM, which suggests that WNTM is more efficient on shorter documents. LF-DMM is the second slowest model on the Snippet dataset. 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 X suggest that GPU-DMM is a desired choice for short text topic modeling, with respect to both effectiveness and efficiency. When computation cost is not a major consideration, GPU-PDMM could deliver more discriminative and coherent topics over short texts.

## 6. CONCLUSION

In this paper, we first propose a new topic model on the basis of the Dirichlet Multinomial Mixture (DMM) model, named PDMM. The difference from DMM is that, in PDMM, each short text can be associated with a small number of relevant topics. The experimental results demonstrate that PDMM delivers better classification accuracy and topic coherence than DMM and other state-of-the-art alternatives in most settings. This suggests that single topic assumption made in many topic model techniques over short texts is inappriporiate in reality. The superiority of PDMM validates that allowing a topic model to infer several relevant topics from a single short text could help better uncover the underlying semantics. We further propose two new topic models on the basis of DMM and PDMM respectively, named GPU-DMM and GPU-PDMM, to leverage global word co-occurrence knowledge to help distil better topics over short texts. Instead of extracting word semantic relations from external thesauri, we propose

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

| Model | Snippet | | | BaiduQA | | |
|---|---|---|---|---|---|---|
| | $K$ = 40 | $K$ = 60 | $K$ = 80 | $K$ = 40 | $K$ = 60 | $K$ = 80 |
| BTM | 0.572 | 0.832 | 1.121 | 1.411 | 2.042 | 2.603 |
| PTM | 1.186 | 1.281 | 1.329 | 11.546 | 12.56 | 13.01 |
| WNTM | 0.609 | 0.88 | 1.181 | 1.172 | 1.706 | 2.243 |
| GMTM | 0.218 | 0.328 | 0.486 | 1.311 | 2.013 | 2.809 |
| LCTM | 18.746 | 18.813 | 18.88 | 94.875 | 95.686 | 96.703 |
| SATM | 0.392 | 0.451 | 0.478 | 10.120 | 10.345 | 11.711 |
| LF-DMM | 5.209 | 7.234 | 9.702 | 13.028 | 20.493 | 31.822 |
| DMM | **0.042** | **0.069** | **0.088** | **0.355** | **0.566** | **0.843** |
| PDMM | 1.6838 | 1.7264 | 1.8464 | 11.639 | 11.748 | 11.826 |
| GPU-DMM | 0.085 | 0.115 | 0.217 | 1.558 | 1.862 | 2.318 |
| GPU-PDMM | 4.527 | 4.706 | 5.445 | 37.788 | 40.687 | 45.203 |

to harness the relations from the results of the recent neural network language model techniques. In GPU-DMM and GPU-PDMM, the generalized Pólya urn (GPU) model is utilized to enhance the topic similarity for two semantically related words which may not co-occur many times in the short text collection. The experimental results based on two real-world short text corpora show that GPU-DMM and GPU-PDMM outperform existing state-of-the-art models in terms of topic coherence and discriminative ability. Nowdays, a larege number of documents are publically available for many languages or domains. By utilizing the word embedding techniques, we believe that many topic models designed for short texts can benefit from incorporating the general semantic word relations via the GPU model. In our study, we also observe that both models are computationally costly than GPU-DMM, DMM and other alternatives. We plan to develop more efficient topic inference strategy for them in the future.

**Acknowledgements.** This research was supported by National Natural Science Foundation of China (No. 61502344), Natural Scientific Research Program of Wuhan University (No. 2042017kf0225,No. 2042016kf0190), Academic Team Building Plan for Young Scholars from Wuhan University (No. Whu2016012) and Singapore Ministry of Education Academic Research Fund Tier 2 (MOE2014-T2-2-066). Chenliang Li is the corresponding author.

## REFERENCES

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. *Neural Probabilistic Language Models*. Springer. 137–186 pages.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research* (2003), 993–1022.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L Boyd-Graber, and David M Blei. 2009. Reading Tea Leaves: How Humans Interpret Topic Models. In *NIPS*. Curran Associates, Inc., 288–296.

Mengen Chen, Xiaoming Jin, and Dou Shen. 2011. Short Text Classification Improved by Learning Multi-Granularity Topics. In *IJCAI*. IJCAI/AAAI, 1776–1781.

Zhiyuan Chen and Bing Liu. 2014. Mining Topics in Documents: Standing on The Shoulders of Big Data. In *SIGKDD*. ACM, 1116–1125.

Zhiyuan Chen, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malú Castellanos, and Riddhiman Ghosh. 2013a. Discovering Coherent Topics using General Knowledge. In *CIKM*. ACM, 209–218.

Zhiyuan Chen, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malú Castellanos, and Riddhiman Ghosh. 2013b. Leveraging Multi-Domain Prior Knowledge in Topic Models. In *IJCAI*. IJCAI/AAAI, 2071–2077.

Xueqi Cheng, Xiaohui Yan, Yanyan Lan, and Jiafeng Guo. 2014. BTM: Topic Modeling over Short Texts. *IEEE Trans. Knowl. Data Eng.* (2014), 2928–2941.

Ronan Collobert and Jason Weston. 2008. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *ICML*. ACM, 160–167.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research* (2011), 2493–2537.

Rajarshi Das, Manzil Zaheer, and Chris Dyer. 2015. Gaussian LDA for Topic Models with Word Embeddings. In *ACL*. The Association for Computer Linguistics, 795–804.

Thomas Hofmann. 1999. Probabilistic Latent Semantic Indexing. In *SIGIR*. ACM, 50–57.

Liangjie Hong and Brian D Davison. 2010. Empirical Study of Topic Modeling in Twitter. In *The First Workshop on Social Media Analytics*. ACM, 80–88.

Liangjie Hong, Dawei Yin, Jian Guo, and Brian D Davison. 2011. Tracking Trends: Incorporating Term Volume into Temporal Topic Models. In *SIGKDD*. ACM, 484–492.

Weihua Hu and Jun'ichi Tsujii. 2016. A Latent Concept Topic Model for Robust Topic Inference Using Word Embeddings. In *ACL*. The Association for Computer Linguistics, 380–386.

Ou Jin, Nathan N Liu, Kai Zhao, Yong Yu, and Qiang Yang. 2011. Transferring Topical Knowledge from Auxiliary Long Texts for Short Text Clustering. In *CIKM*. ACM, 775–784.

Tom Kenter and Maarten de Rijke. 2015. Short Text Similarity with Word Embeddings. In *CIKM*. ACM, 1411–1420.

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. From Word Embeddings To Document Distances. In *ICML*. JMLR.org, 957–966.

Chenliang Li, Haoran Wang, Zhiqian Zhang, Aixin Sun, and Zongyang Ma. 2016. Topic Modeling for Short Texts with Auxiliary Word Embeddings. In *SIGIR*. ACM, 165–174.

Zongyang Ma, Aixin Sun, Quan Yuan, and Gao Cong. 2012. Topic-Driven Reader Comments Summarization. In *CIKM*. ACM, 265–274.

Hosam Mahmoud. 2008. Polya Urn Models. *Chapman & Hall/CRC Texts in Statistical Science* (2008).

Rishabh Mehrotra, Scott Sanner, Wray Buntine, and Lexing Xie. 2013. Improving LDA Topic Models for Microblogs via Tweet Pooling and Automatic Labeling. In *SIGIR*. ACM, 889–892.

Tomas Mikolov, Kai Chen, Greg Corrada, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781* (2013).

David Mimno, Hanna M Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing Semantic Coherence in Topic Models. In *EMNLP*. The Association for Computational Linguistics, 262–272.

Andriy Mnih and Geoffrey E Hinton. 2009. A Scalable Hierarchical Distributed Language Model. In *NIPS*. Curran Associates, Inc., 1081–1088.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic Evaluation of Topic Coherence. In *HLT-NAACL*. The Association for Computational Linguistics, 100–108.

Dat Quoc Nguyen, Richard Billingsley, Lan Du, and Mark Johnson. 2015. Improving Topic Models with Latent Feature Word Representations. *Transactions of the*

*Association for Computational Linguistics* (2015), 299–313.

Alexandru Niculescu-Mizil and Rich Caruana. 2005. Predicting Good Probabilities with Supervised Learning. In *ICML*. ACM, 625–632.

Kamal Nigam, Andrew Kachites MacCallum, Sebastian Thrun, and Tom Mitchell. 2000. Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning* (2000), 103–134.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*. The Association for Computational Linguistics, 1532–1543.

Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi. 2008. Learning to Classify Short and Sparse Text & Teb with Hidden Topics from Large-Scale Data Collections. In *WWW*. ACM, 91–100.

Ian Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling. 2008. Fast Collapsed Gibbs Sampling for Latent Dirichlet Allocation. In *SIGKDD*. ACM, 569–577.

Xiaojun Quan, Chunyu Kit, Yong Ge, and Sinno Jialin Pan. 2015. Short and Sparse Text Topic Modeling via Self-Aggregation. In *AAAI*. AAAI Press, 2270–2276.

Daniel Ramage, Susan T Dumais, and Daniel J Liebling. 2010. Characterizing Microblogs with Topic Models.. In *ICWSM*. The AAAI Press, 130–137.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1988. Neurocomputing: Foundations of Research. (1988), 696–699.

Vivek Kumar Rangarajan Sridhar. 2015. Unsupervised Topic Modeling for Short Texts Using Distributed Representations of Words. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing, VS@NAACL-HLT 2015, June 5, 2015, Denver, Colorado, USA*. The Association for Computational Linguistic-s, 192–200.

Bharath Sriram, Dave Fuhry, Engin Demir, Hakan Ferhatosmanoglu, and Murat Demirbas. 2010. Short Text Classification in Twitter to Improve Information Filtering. In *SIGIR*. ACM, 841–842.

T.P. Straatsma, H.J.C. Berendsen, and A.J. Stam. 1986. Estimation of Statistical Errors in Molecular Simulation Calculations. *Molecular Physics* 57, 1 (1986), 89–95.

Aixin Sun. 2012. Short Text Classification using Very Few Words. In *SIGIR*. ACM, 1145–1146.

Chong Wang and David M Blei. 2011. Collaborative Topic Modeling for Recommending Scientific Articles. In *SIGKDD*. ACM, 448–456.

Xuanhui Wang, ChengXiang Zhai, Xiao Hu, and Richard Sproat. 2007. Mining Correlated Bursty Topic Patterns from Coordinated Text Streams. In *SIGKDD*. ACM, 784–793.

Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. 2010. Twitterrank: Finding Topic-sensitive Influential Twitterers. In *WSDM*. ACM, 261–270.

Xiaohui Yan, Jiafeng Guo, Yanyan Lan, and Xueqi Chen. 2013. A Biterm Topic Model for Short Texts. In *WWW*. International World Wide Web Conferences Steering Committee / ACM, 1445–1456.

Jianhua Yin and Jianyong Wang. 2014. A Dirichlet Multinomial Mixture Model-based Approach for Short Text Clustering. In *SIGKDD*. ACM, 233–242.

Xia Yunqing, Tang Nan, Hussain Amir, and Cambria Erik. 2015. Discriminative Bi-Term Topic Model for Headline-Based Social News Clustering. In *AAAI*. AAAI Press, 311–316.

Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. 2011. Comparing Twitter and Traditional Media using Topic Models. In *ECIR*. Springer, 338–349.

Guoqing Zheng and Jamie Callan. 2015. Learning to Reweight Terms with Distributed

Representations. In *SIGIR*. ACM, 575–584.

Yuan Zuo, Junjie Wu, Hui Zhang, Hao Lin, Fei Wang, Ke Xu, and Hui Xiong. 2016a. Topic Modeling of Short Texts: A Pseudo-Document View. In *KDD*. ACM, 2105–2114.

Yuan Zuo, Jichang Zhao, and Ke Xu. 2016b. Word Network Topic Model: A Simple but General Solution for Short and Imbalanced Texts. *Knowl. Inf. Syst.* 48, 2 (2016), 379–398.