ATLANTIS PRESS
Advances in Engineering Research, volume 155
2nd International Conference on Advances in Energy, Environment and Chemical Science (AEECS 2018)

Topic Model over Short Texts Incorporating Word Embedding

Kai Yu [1, a], Yiming Zhang [1] and Xu Wang [1]
[1] College of Computer, National University of Defense Technology, Changsha, Hunan
[a]yukaigoforit@126.com

Keywords: Short texts, Topic model, Word embedding, Text mining

Abstract. Short texts' data sparsity makes them difficult to find out their document-level word co-occurrence patterns, that's why conventional topic models like LDA experience a large performance degradation over short texts. As a derivative product of learning neuro probabilistic language model, word embedding can well express semantic similarity of word. In this paper, we propose a new model called promotion-BTM, which promotes the probability that similar words based on word embedding belong to the same topic. It also distinguishes the words of a biterm into topical word and general word, and only promotes topical words' semantically similar words. Extensive experiments on real-world datasets show that our model exceeds the baseline model BTM on all evaluations.

1. Introduction

Short texts are widely distributed on internet, including search snippets, tweets, news title and so on. Text mining has been used in many tasks, such as recommendation, sentiment classification, topic detection and tracking. Topic model is a method for mining latent semantics of texts. Conventional topic models, pLSA (probabilistic Latent Semantic Analysis) [1] and LDA (Latent Dirichlet Allocation) [2] have been widely used in mining document. They infer these two distributions by capturing document-level word co-occurrence patterns, but they suffer a lot on short texts due to the data sparsity of short texts.

In recent years, many researchers try to weaken the effects of data sparsity in short texts and adopt some methods to assist the statistics of word co-occurrence patterns. Phan et al. inferred short texts' hidden topics with distributions learnt on external corpus for text classification [3]. Some researchers aggregated all tweets in a period of a user [4], or aggregated tweets by their hashtags or timestamps [5] and revealed their topics. Paper [8] promoted a self-aggregation topic model based on topical similarity, it used similar process as LDA to generate pseudo documents and extracted short texts from each document. There are some works directly using word co-occurrence networks, and represent each word with its co-occurrent words [9].

Among these methods, *biterm topic model* (BTM) proposed by Yan et al. is very impressed [10]. It directly models the word co-occurrence pattern as a biterm (an unordered word pair), and reveals topics by modeling generation process of the whole corpus' biterms, rather than each document's words. This way effectively weakens short texts' sparse problem. It has been demonstrated BTM can learn more coherent topics than LDA on short texts. But BTM also has some shortcomings, for example, it doesn't have ability to distinguish topical word and general word, and the hypothesis that both words of a biterm belong to the same topic is too strong. So there's still opportunity to improve BTM and we choose it as the main starting point of this paper.

Word embedding (also called word vector) is a kind of distributed representation for word and has been playing an important role in NLP (natural language processing) tasks. As a derivative product of neuro probabilistic language model, word embedding implies words' semantic and syntactic information in each dimension of vector, and well expresses semantic similarity. In our human knowledge, semantically similar words are more likely to occur in the same topic, this inspires us to incorporate word embedding into topic model to help revealing short texts' latent topics.

In this paper, we propose a new topic model promotion-BTM, by incorporating topic model with word embedding. The main alternations based on BTM are summarized as follows:

1. We increase the probability that semantically similar words based on word embedding belong to the same topic by also updating the frequency that similar words of current sampled word occur in

the same sampled topic in the process of inferring model parameters with Gibbs sampling, it can be seen as promotion.

2. In the generation process of the whole corpus' biterms, the hypothesis that the both words of a biterm belong to the same topic is too strong. Therefore, we distinguish a word into topical word and general word judged by its topic probability distribution. Meanwhile only topical word's similar words can be promoted.

We evaluate our new model's performance on two real-word datasets. Experiment results demonstrate our model's superiority compared to baseline methods, including LDA and BTM.

2. Proposed Method

In this section, we first give a brief introduction about BTM, which is basics of our work. Then we present how to promote the probability that semantically similar words belong to the same topic. Finally we introduce in detail our topic model promotion-BTM and its parameters inference process.

Biterm Topic Model. A biterm represents two distinct unordered words in a context, it is just an instance of word co-occurrence patterns. BTM directly models generation process of the whole corpus' biterms, i.e. the whole corpus have many topics and each biterm is extracted from one topic. Shown in Fig. 1 (a), the generation process of biterms in BTM can be described as follows:

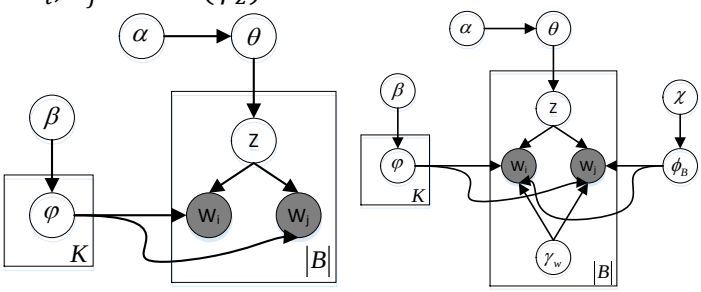
(1) Draw the whole corpus' topic distribution $\theta \sim \mathrm{Dir}(\alpha)$

(2) For b in biterms do

a) Draw a topic $z \sim \mathrm{Multi}(\theta)$

b) Draw topic z's word distribution $\phi_z \sim Dir(\beta)$

c) Draw two words $w_i, w_j \sim Multi(\phi_z)$



(a) (b)

Fig. 1. Graphical model of BTM (a) and promotion-BTM (b)

Method of Promotion. We compute cosine similarity of words by their vectors. Word embedding's quality affects the similarity measurement, fortunately word2vec [6] and glove [7] provide fine pre-trained word embedding that we can adopt directly. If the similarity of two words is bigger than a preset threshold τ, we consider they are similar and record them both. But if a word has too much similar words, more than the preset threshold value N_{filter}, we think it is too common and discard it from the records. E.g. measure words, like 'vast', 'mass', 'plenty' all means 'a lot'. If a word in corpus doesn't exist in pre-trained word embedding, we represent it with zero vector and assume it has no similar words.

In LDA, according to the character of Dirichlet distribution, the topic-word multinomial distribution ϕ is finally computed by the count that words occur in each topic and the Dirichlet prior hyper-parameter β. The same goes for documents-topic multinomial distribution θ. As shown in Eq.1.

$$\theta_d^k = \frac{n_d^k + \alpha_k}{\sum_{k \in K} n_d^k + \alpha_k}, \quad \phi_k^w = \frac{n_k^w + \beta_w}{\sum_{w \in W} n_k^w + \beta_w} \tag{1}$$

θ_d^k denotes the probability of topic k occurs in document d, n_d^k denotes the frequency of topic k occurs in document d. ϕ_k^w denotes the probability of word w occurs in topic k. n_k^w denotes the frequency of word w occurs in topic k.

n_d^k and n_k^w are updated during the Gibbs sampling process, that is, at each iteration for sampling a new topic assignment to a word, if new topic k^* is sampled, then augment the counts on new topic, and the counts on old topic k is decreased, as shown in Eq.2.

$$n_d^{k^*} \leftarrow n_d^{k^*} + 1,\ n_{k^*}^w \leftarrow n_{k^*}^w + 1\ ,\ n_d^k \leftarrow n_d^k - 1,\ n_k^w \leftarrow n_k^w - 1 \tag{2}$$

So our approach for promotion is to do something to the count, n_d^k and n_k^w. Apart from the regularly update for sampled word, we also update the count for similar words of sampled word. In detail, for each similar word w^* of word w, we do additional update as shown in Eq.3 and Eq.4.

$$n_d^{k^*} \leftarrow n_d^{k^*} + Similarity_{w,w^*} \times weight\ ,\ n_{k^*}^w \leftarrow n_{k^*}^w + Similarity_{w,w^*} \times weight \tag{3}$$

$$n_d^k \leftarrow n_d^k - Similarity_{w,w^*} \times weight\ ,\ n_k^w \leftarrow n_k^w - Similarity_{w,w^*} \times weight \tag{4}$$

$Similarity_{w,w^*}$ denotes cosine similarity between two words, *weight* is a preset value, represents how much we want to promote the similar words. This additional update can be seen that we also sample new topic assignment for similar words, and the new topic is just the same as current word's. Thus the probability that semantically similar words belong to the same topic is promoted.

When current word is not relevant to the newly sampled topic, it will impair the topic learning if we still promote similar words, because it results in an enhanced probability these words occur in irrelevant topics. Therefore, a word needs to be distinguished into topical word and general word, we can judge by this word's topic probability distribution P(z|w), which can be computed by topic distribution P(z) and topic-word distribution P(w|z), through Bayes formula as shown in Eq.5.

$$p(z|w) = \frac{p(z)\,p(w|z)}{\sum_{i=1}^{K} p(i)\,p(w|i)} \tag{5}$$

If the probability that a word occur in a specific topic is in the top among all topics, the word can be considered highly relevant to the topic. It could be described as rate λ in Eq.6. We draw a random number μ in [0, 1], if μ ≥ λ, then word w is judged as topical word, otherwise as general word. Obviously, higher the probability is, smaller λ is, and more possible that μ is bigger than λ, so the word is more likely judged as a topical word.

$$\lambda = \frac{\max_{k \in z} p(z=k|w) - p(z|w)}{\max_{k \in z} p(z=k|w)} \tag{6}$$

The promotion-BTM. Because the hypothesis that both words of a biterm belong to the same topic is too strong, we distinguish topical word and general word in the generation process of biterms. As shown in Fig.1 (b), promotion-BTM add a new Dirichlet-Multinomial conjugation based on BTM. χ is the Dirichlet prior of general word distribution ϕ_B. An indicator γ_w is used to indicate word w is drawn from topic-word distribution or general word distribution. γ_w can be computed with the method in Eq.6 as mentioned above. The generation process of biterms in promotion-BTM can be described as follows:

(1) Draw the whole corpus' topic distribution θ~Dir(α)
(2) Draw the corpus' general word distribution ϕ_B~Dir(χ)
(3) For b in biterms do
a) Draw a topic z~Multi(θ)
b) Draw topic z's word distribution $\phi_z \sim Dir(\beta)$
c) For w_i, w_j in b do
i. Compute μ = Random(0,1) and rate λ in Eq.6
ii. If μ ≥ λ, draw a word w~$Multi(\phi_z)$
iii. If μ < λ, draw a word w~$Multi(\phi_B)$

After two above alterations, Gibbs sampling for our new model can be shown in Algorithm 1. Similarly as Eq.1 in LDA, for promotion-BTM, corpus-topic distribution θ_z can be computed with count n_z, topic-word distribution ϕ_z^w is computed with topical word distribution and general word distribution as shown is Eq.7 and each can be compute with count n_z^w and n_{bg}^w respectively.

$$\theta_z = \frac{n_z + \alpha}{\sum_{z \in Z} n_z + \alpha}, \quad \phi_z^w = \gamma_w \frac{n_z^w + \beta}{\sum_{w \in W} n_z^w + \beta} + (1 - \gamma_w) \frac{n_{bg}^w + \chi}{\sum_{w \in W} n_{bg}^w + \chi} \tag{7}$$

Algorithm 1: Gibbs sampling for promotion-BTM

Input: topic number K, Dirichlet prior α, β, χ, all biterms B, similar words and their similarity SW, promotion weight.
Output: topic distribution θ and topic-word distribution ϕ.
Algorithm:
Step 1: assign topics randomly for all biterms.
Step 2: count n_z, n_z^w, n_{bg}^w.
Step 3:
For iter = 0,1,2⋯ Repeat
  Compute P(z), P(w|z), P(z|w)
  For b in B do
    For w in b do
      $n_z \leftarrow n_z - 1$, $n_z^w \leftarrow n_z^w - 1$
      For w^* in SW_w do
        Do updates as in Eq.4
    For w in b do
      Compute μ = Random(0,1) , rate λ in Eq.6
      If $\mu \geq \lambda$: $\gamma_w = 1$, $n_{bg}^w \leftarrow n_{bg}^w - 1$
      If $\mu < \lambda$: $\gamma_w = 0$, $n_{bg}^w \leftarrow n_{bg}^w + 1$
    Sample a new topic $z' \sim P(z|z_{\neg b}, B, \alpha, \beta, \chi)$
    For w in b do
      $n_{z'} \leftarrow n_{z'} + 1$, $n_{z'}^w \leftarrow n_{z'}^w + 1$
      For w^* in SW_w do
        Do updates as in Eq.3
Step 4: compute topic distribution θ and topic-word distribution ϕ

We can obtain the conditional probability $P(z|z_{\neg b}, B, \alpha, \beta, \chi)$ as shown in Eq.8 similarly as BTM. The inference for document-topic distribution is the same as in BTM.

$$p\left(z \middle| z_{\neg b}, B, \alpha, \beta, \chi\right) = \frac{P(z) P\left(w_i \middle| z\right) P\left(w_i \middle| z\right)}{\sum_z P(z) P\left(w_i \middle| z\right) P\left(w_i \middle| z\right)} \propto \theta_z \cdot \phi_z^{w_i} \cdot \phi_z^{w_j} \tag{8}$$

3. Experiment

Datasets.

Training corpus. We choose two English labeled short texts corpus, TagMyNews, WebSnippets. TagMyNews is a dataset of English news extracted from RSS feeds of popular news websites, we extract the news title as our short texts. WebSnippets is a dataset created by Phan et al. for their work on short texts classification [3]. It is composed by search snippets drawn from Google. We do some preprocessing for all datasets as follows: (1) All words are down-cased. (2) Remove all punctuations and English stop words found in the stop words list of NLTK toolkit. (3) Remove all the words whose document frequency is less than 5. The details about our processed datasets are listed in Table. 1.

Table.1 Dateset details

Dataset	Vocabulary	Doc number	Doc length	Categories
TagMyNews	24,641	32,602	6.48	7
WebSnippets	30,633	12,340	7.90	8

Word embedding. Google word2vec [6] and Stanford glove [7] are most two popular open source toolkit to train word embedding. We directly download these two pre-trained word embeddings. Word2vec's embedding is trained on part of Google News dataset (about 100 billion words), which

contains 300-dimensional vectors for 3 million words and phrases. Glove's is trained on common crawled web dataset (about billion words and phrases), which has a dimension of 300 and a size of 2 million words. We use these two word embeddings to construct semantically similar words and name the new models promotion-BTM_w2v and promotion-BTM_glo respectively.

Experimental Setup. In our all experiments, we set topic number K=20 (40, 60, 80) respectively and Dirichlet prior α=50/K, β=0.01 for all models, set another Dirichlet prior χ=0.01 for promotion-BTM. We set iteration number for Gibbs sampling equals 100. We run each experiment for 3 times and compute the metrics' mean value as our final results. When constructing similar words based on word embedding, we need to set up the threshold value of cosine similarity τ. To determine the value, we sample many words in word2vec's pre-training word embedding, and check their top 50 most similar words, then analysis their similarity, finally we set τ=0.5. If a word has more than N_{filter} similar words, we discard it from the records. We experimentally set N_{filter} = 50.

Evaluation Metrics. The goal of topic model is to learn two multinomial distributions of corpus, document-topic distribution θ and topic-word distribution ϕ. The usual way to evaluate topic model is to judge these two distributions' facticity. The former is evaluated by topic coherence, means how much relevant the topical words of a topic is. The latter means how much document is relevant to the learnt topics, which can be assessed indirectly with document classification and document clustering.

Topic Coherence. To assess the topic coherence, we obtain top 15 high probable words under each topic, compute the co-occurrence frequency in an external corpus for each two words. As shown in Eq.9, we use NPMI-Score (normalized pointwise mutual information) to compute the coherence, which has a strong correlation with human judged coherence. We take the average value of all topics as a model's topic coherence score. Smaller the NPMI-score is, more topical coherent the model is. We use the English Wikipedia of 4.6 million articles as our external corpus.

$$NPMI-Score=\frac{\log\frac{p(w_i,w_j)}{p(w_i)p(w_j)}}{\log p(w_i)p(w_j)} \tag{9}$$

Document classification. The learnt topic distribution for a document can be seen as its feature vector, and it can be used to classify the document. In our experiments, we use linear-SVM as our classifier and F1-Score as classification metrics. We do document classification on each distribution for 5 times and compute their mean F1-Score as final result.

Document clustering. We also do document clustering on document-topic distribution. We use k-means algorithm to cluster all documents and compute clusters' purity to assess clustering effect. Purity measures the proportion that correctly clustered samples account for. It takes the dominant category in a cluster as this cluster's category, as shown in Eq.10.

$$purity(\Omega,C)=\frac{1}{N}\sum_k \max_j \left|\omega_k \cap c_j\right| \tag{10}$$

Experiment results. We promote similar words of sampled word based on similarity and a preset value *weight*, as shown in Eq.3 and Eq.4. To find out the best value of promotion weight, we set *weight* equals 0.5 to 1 at a step of 0.1, and K=40 to evaluate model's performance on TagMyNews dataset. We find promotion-BTM show the best performance when weight equals 0.5 or 0.6. Considering most words' similarity is between 0.5 and 0.7, the actual updates on count n_z^w and n_z are just no more than 0.4. This value is reasonable, because our model doesn't actually sample new topics for these similar words. Similar words will be over promoted if we set the update value equals 1, and it will impair model's performance as the curve goes down. So we set promotion weight equals 0.5 in our subsequent experiments.

We do all three evaluations mentioned above on three datasets. The results on NPMI-Score, F1-Score and purity are listed at TABLE.II. Results show that our new model gains improvement on all evaluation metrics in different degrees. BTM indeed shows better performance than conventional topic model LDA. Our new model can learn more coherent topics than baseline model BTM, which demonstrates effectiveness of our strategy by incorporating word embedding into topic model. Furthermore we also can see that word2vec and glove's pre-trained word embeddings are neck and

neck in our model. The topic number also affect model's topic coherence, all metrics rise when K increases, but when it gets bigger than 80, topic model's performance gets worse.

Table.2 Evaluations on two datasets

Dataset	Model	NPMI-Score				F1-Score				Purity			
		K20	*K40*	*K60*	*K80*	*K20*	*K40*	*K60*	*K80*	*K20*	*K40*	*K60*	*K80*
Web Snippets	LDA	14.75	14.35	14.94	15.74	0.50	0.60	0.64	0.62	0.55	0.60	0.62	0.62
	BTM	12.96	12.27	12.88	13.16	0.52	0.63	0.64	0.64	0.56	0.61	0.64	0.64
	pBTM_w2v	11.38	10.39	11.08	11.74	**0.57**	0.64	**0.65**	0.65	**0.57**	0.63	**0.65**	**0.68**
	pBTM_glo	**10.69**	**9.80**	**10.98**	**11.37**	0.56	**0.64**	0.65	**0.65**	0.57	**0.63**	0.65	0.67
Improvement		2.27	2.47	1.89	1.79	0.05	0.02	0.01	0.01	0.01	0.02	0.01	0.03
Tagmy news	LDA	15.93	15.24	15.14	15.44	0.55	0.61	0.62	0.61	0.56	0.59	0.59	0.59
	BTM	13.76	13.46	12.96	13.56	0.56	0.64	0.64	0.64	0.59	0.64	0.65	0.64
	pBTM_w2v	**8.41**	**7.42**	**8.01**	9.60	0.56	0.66	0.67	**0.67**	0.63	0.65	0.66	**0.67**
	pBTM_glo	8.81	8.01	**8.01**	**9.00**	**0.58**	**0.66**	**0.67**	0.67	**0.64**	**0.66**	**0.67**	0.67
Improvement		5.34	6.03	4.95	4.55	0.02	0.02	0.02	0.03	0.05	0.02	0.02	0.03

4. Conclusions

Mining short texts has been applied in various fields and brings significant benefits to people. While short texts' data sparsity problem makes it difficult for conventional topic model to reveal their latent semantic. In this paper, we propose a strategy of incorporating topic model with word embedding and applied it to BTM and proposed promotion-BTM. It increases the probability that similar words based on word embedding belong to the same topic, and also distinguishes word into topical word and general word. We carry out empirical studies on real-world datasets, experiment results show that our model exceeds the baseline model BTM on all evaluations, by topic coherence and documents classification and documents clustering.

References

[1] Hofmann T. Probabilistic latent semantic indexing[C]// International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, 1999:50-57.

[2] Blei D M, Ng A Y, Jordan M I. Latent dirichlet allocation[J]. Journal of Machine Learning Research, 2003, 3:993-1022.

[3] Phan X H, Nguyen L M, Horiguchi S. Learning to classify short and sparse text & web with hidden topics from large-scale data collections[C]//Proceedings of the 17th international conference on World Wide Web. ACM, 2008: 91-100.

[4] Weng J, Lim E P, Jiang J, et al. Twitterrank: finding topic-sensitive influential twitterers[C]//Proceedings of the third ACM international conference on Web search and data mining. ACM, 2010: 261-270.

[5] Hong L, Davison B D. Empirical study of topic modeling in twitter[C]//Proceedings of the first workshop on social media analytics. ACM, 2010: 80-88.

[6] Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality[C]//Advances in neural information processing systems. 2013: 3111-3119.

[7] Pennington J, Socher R, Manning C D. Glove: Global Vectors for Word Representation[C]//EMNLP. 2014, 14: 1532-1543..

[8] Quan X, Kit C, Ge Y, et al. Short and Sparse Text Topic Modeling via Self-Aggregation[C]//IJCAI. 2015: 2270-2276.

[9] Zuo Y, Zhao J, Xu K. Word network topic model: a simple but general solution for short and imbalanced texts[J]. Knowledge and Information Systems, 2016, 48(2): 379-398.

[10] Yan X, Guo J, Lan Y, et al. A biterm topic model for short texts[C]//Proceedings of the 22nd international conference on World Wide Web. ACM, 2013: 1445-1456.