Knowledge-Based Systems 182 (2019) 104812

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Topic model with incremental vocabulary based on Belief Propagation☆

Meng Wang*, Lu Yang, JianFeng Yan, Jianwei Zhang, Jie Zhou, Peng Xia

School of Computer Science & Technology, Soochow University, Su Zhou, Jiang Su, China

ARTICLE INFO

Article history:
Received 26 October 2018
Received in revised form 24 June 2019
Accepted 24 June 2019
Available online 31 July 2019

Keywords:
Topic model
Belief Propagation
Stick-breaking process
Online algorithm

ABSTRACT

Most of the LDA algorithms make the same limiting assumption based on a fixed vocabulary. When these algorithms process data streams in real time, the non-existent words in the vocabulary are discounted. Unexpected words that appear in the streams are incapable to be processed, as the atoms in the Dirichlet distribution are fixed. In order to address the drawbacks as mentioned above, ivLDA with topic–word distribution stemming from the Dirichlet process that has infinite atoms instead of Dirichlet distribution is proposed. ivLDA involves an incremental vocabulary that enables the topic models to process data streams. Besides, two methods are presented to manage the indices of the words, namely, ivLDA-Perp and ivLDA-PMI. ivLDA-Perp is capable of achieving high accuracy and ivLDA-PMI is able to identify the most valuable words to represent the topic. As indicated by experiments, ivLDA-Perp and ivLDA-PMI can achieve superior performance to infvoc-LDA and other state-of-the-art algorithms with fixed vocabulary.

© 2019 Elsevier B.V. All rights reserved.

1. Introduction

Latent Dirichlet Allocation (LDA) [1,2] is known as a matrix factorization algorithm that decomposes the document–word matrix into the document–topic matrix and the topic–word matrix. Batch LDA algorithms [3] represented by Variational Bayes (VB) [1], Gibbs Sampling (GS) [4] and Belief Propagation (BP) [5] all display a very high level of space complexity. As the size of information sources grows on a continued basis in recent years, these algorithms tent to have strict requirements on computer memory. Therefore, for the handling of a large corpus, applying online LDA algorithms [6,7] is a more efficient choice. Online LDA algorithms eliminate the need to save the whole document–word matrix into memory. They split the corpus into batches and then make incremental estimate of the distribution. Each batch is discarded from memory after the algorithms complete processing of it. Most online algorithms are derived from their batch versions, like Online Variational Bayes (OVB) [8,9], Online Gibbs Sampling (OGS) [3,10,11] and Online Belief Propagation (OBP) [12]. There are two major differences between them, one being that they are based on different variants of LDA and the other being that they have different approaches to refreshing parameters.

In our view, the most valuable product of LDA lies in the topic–word distribution. Nevertheless, a large proportion of the online LDA algorithms make the same limiting assumption based on a fixed vocabulary, implying that the words we train are determined at a time when the process is initiated. When algorithms with fixed vocabulary process data streams in real time, unexpected words that appear in streams cannot be processed, as the atoms in the Dirichlet distribution are fixed, which means words excluded from the vocabulary are discounted and new words cannot be added. There are usually two solutions to LDA algorithms with fixed vocabulary: (1) Making a self-satisfied vocabulary. The words excluded from the vocabulary are ignored. Despite being a common solution, it can cause the loss of crucial information. (2) Making a very large vocabulary to ensure that all words in streams are included. However, it places a big burden on memory when the vocabulary is overly large.

In order to address the drawbacks as mentioned above, the ivLDA algorithm is suggested. The topic–word distribution of ivLDA follows the Dirichlet process (DP) [13,14], which has infinite atoms, not the Dirichlet distribution. ivLDA has an incremental vocabulary which does not permit the addition of new words. Despite that the vocabulary can be made flexible by using DP, DP could give rise to some issues. Considering that DP is an infinite version of the Dirichlet Allocation, so we make efforts to ensure that DP has a truncation method. The order of atoms in DP is essential, and the words with lower indices may have a

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.knosys.2019.06.020.

* Corresponding author.
E-mail address: mwang030@stu.suda.edu.cn (M. Wang).

https://doi.org/10.1016/j.knosys.2019.06.020
0950-7051/© 2019 Elsevier B.V. All rights reserved.

higher probability than the ones with higher indices, for which we design two methods to manage the indices of words, ivLDA-Perp and ivLDA-PMI. ivLDA-Perp can achieve high accuracy and ivLDA-PMI can identify the most valuable words to represent the topic.

In recent years, very few LDA algorithms with infinite vocabulary have been proposed, the most representative of which is infvoc-LDA [15]. infvoc-LDA was suggested by Ke Zhai in 2013, who assumed the topic–word distribution stems from DP. Our models are noticeably different from theirs:

(1) infvoc-LDA is premised on the structure of SOI [16], but ivLDA is strictly based on BP. The basements of the two algorithms are starkly different.
(2) There are two parameters of DP, one of which is the base distribution (G0). The base distribution of ivLDA is the uniform distribution, which suggests that every word in topic k gets the identical probability from the stick-breaking construction. The base distribution of infvoc-LDA bears similarity to the N-gram model, which indicates that each word in topic k has a different probability.

The details of the differences between ivLDA and infvoc-LDA are described in the Section 3.

The remainder of this paper is structured as follows: The LDA algorithm, the BP algorithm, and other related work are reviewed in Section 2. In Section 3, the ivLDA algorithm is derived and ivLDA-Perp and ivLDA-PMI are described in detail. A comparison is performed of ivLDA with infvoc-LDA and other state-of-the-art algorithms with fixed vocabulary in Section 4. Section 5 draws conclusions and indicates envisions for future work.

2. Background

Latent Dirichlet Allocation (LDA) [1,2] is a graphical topic model, which has a three-level structure: corpus level, document level, and word level. It assumes that each document in corpus can be characterized by a particular set of topics. The main task of LDA is to assign a topic label for each word in each document. In LDA, each document d in corpus can be represented by a document–topic distribution θ_d. The number of atoms of distribution θ_d is K, where K denotes the number of topics. Each topic k can be represented by a topic–word distribution ϕ_k. The atoms of distribution ϕ_k are the words in the vocabulary.

The generative process of LDA is as follows: Sample from the Dirichlet distribution with a symmetric parameter α to get the distribution of topics in document d. Sample from the Dirichlet distribution θ_d with a symmetric parameter β to get the distribution of words ϕ_k in topic k. After that, choose a topic k for word i in document d from θ_d, choose a word w_i from ϕ_k. Repeat the process above until all documents are generated.

The joint probability of LDA [1] is:

$$\mathrm{p}\left(\mathrm{x}, \mathrm{z}, \theta, \phi|\alpha, \beta\right) \propto \prod_{k=1}^{K} p\left(\phi_k|\beta\right) \prod_{d=1}^{D} p\left(\theta_d|\alpha\right) \prod_{d=1}^{D} \prod_{i=1}^{N_d} p\left(w_i|z_i^k, \phi_k\right) p\left(z_i^k|\theta_d\right) \tag{1}$$

2.1. Belief propagation

Integrating out the multinomial parameters θ_d and ϕ_k in (1), we obtain the joint probability of the collapsed LDA [4]:

$$\begin{aligned}\mathrm{p}\left(\mathrm{x}, \mathrm{z}|\alpha, \beta\right) \propto & \prod_d \prod_k \Gamma\left(\sum_w x_{w,d} z_{w,d}^k + \alpha\right) \\ & \times \prod_w \prod_k \Gamma\left(\sum_d x_{w,d} z_{w,d}^k + \beta\right) \\ & \times \prod_k \Gamma\left(\sum_{w,d} x_{w,d} z_{w,d}^k + W\beta\right)^{-1}\end{aligned} \tag{2}$$

where $\Gamma(\cdot)$ is the gamma function, α and β are the hyperparameters. To maximize the likelihood of z in (2), the BP algorithm provides exact solutions. In BP, the conditional joint probability $p\left(z_{w,d}^k = 1|z_{-(w,d)}^k, x\right)$, called message $\mu_{w,d}(k)$, $0 \le \mu_{w,d}(k) \le 1$, can be normalized by $\sum_{k=1}^{K} \mu_{w,d}(k) = 1$. The message update equation is:

$$\mu_{w,d}(\mathrm{k}) \propto \frac{\left(\hat{\theta}_{-w,d}(k) + \alpha\right) \times \left(\hat{\phi}_{w,-d}(k) + \beta\right)}{\hat{\phi}_{-(w,d)}(k) + W\beta} \tag{3}$$

where the sufficient statistics are:

$$\hat{\theta}_{-w,d}(k) = \sum_{-w} x_{w,d} \mu_{w,d}(\mathrm{k}) \tag{4}$$

$$\hat{\phi}_{w,-d}(k) = \sum_{-d} x_{w,d} \mu_{w,d}(\mathrm{k}) \tag{5}$$

where $-w$ and $-d$ denote all words except w and all documents except d respectively. We can find that message $\mu_{w,d}(k)$ depends on the neighboring message $\mu_{-(w,d)}(k)$. The topic distribution θ_d of document d and the word distribution ϕ_k of topic k can be estimated from the sufficient statistics $\hat{\theta}_d(k)$ and $\hat{\phi}_w(k)$ by Dirichlet normalization:

$$\theta_d(k) = \frac{\hat{\theta}_d(k) + \alpha}{\sum_k \hat{\theta}_d(k) + K\alpha} \tag{6}$$

$$\phi_w(k) = \frac{\hat{\phi}_w(k) + \beta}{\sum_w \hat{\phi}_w(k) + W\beta} \tag{7}$$

BP will iterate Eq. (3) until $\hat{\theta}_d(k)$ and $\hat{\phi}_w(k)$ converge. More details you can find in [5].

As we mentioned above, BP saves all documents in memory and scans them over and over again until the algorithm converges. When the corpus is too large, the cost of memory will be very large. To overcome the drawback of BP, Zeng proposed Online Belief Propagation (OBP) [12,14,17,18]. In OBP, corpus is spilt into batches $x_{w,d}^s$, d ∈ [1, D_s], w ∈ [1, ∞], s ∈ [1, ∞],where D_s is the number of documents in the current batch and s is the index of this batch. Each batch is freed from memory after the processing is complete. Products of the batch in memory are the message matrix $\mu_{K\times NNZ_x}$ and the topic parameter matrix $\hat{\theta}_{K\times D_s}$, which depend only on the current batch, will also be freed too. The global parameter matrix $\phi_{K\times W}$ will remain in memory until the algorithm ends. OBP initialize the messages randomly, and then initialize the sufficient statistics $\hat{\theta}_d^s(k)$ and $\hat{\phi}_w^s(k)$:

$$\hat{\theta}_d^s(k) = \sum_w x_{w,d}^s \mu_{w,d}^s(\mathrm{k}) \tag{8}$$

$$\hat{\phi}_w^s(k) = \hat{\phi}_w^{s-1}(k) + \sum_d x_{w,d}^s \mu_{w,d}^s(\mathrm{k}) \tag{9}$$

where $\hat{\phi}_w^{s-1}(k)$ is the sufficient statistic of the previous batch. After the initialization is completed, OBP will run BP on every

batch. After finishing running the last batch in memory, $\hat{\phi}^{s}_{w}(k)$, the products of BP, fits all the batches that have already been processed.

2.2. Dirichlet process

The Dirichlet process [14,19] is a stochastic process using Bayesian non-parametric methods. It can be seen as the infinite-dimensional generalization of the Dirichlet distribution. DP is just a concept and there are some approaches to implement it such as Chinese restaurant process [20] and Stick-breaking process [13]. We just detail the stick-breaking process, because it is the simplest approach to construct DP.

In the Stick-breaking process view, DP is a distribution with two parameters, denoted DP(α, G_0), where α is the positive scalar parameter of Beta distribution, G_0 is the base distribution of the DP. If $\pi \sim$ DP(α, G_0), π is constructed according to the following functions:

$$\pi = \sum_{i=1}^{\infty} V_i \prod_{j=1}^{i-1} (1 - V_j)\, \delta_{\theta_i} \tag{10}$$

$$V_i \sim Beta(1, \alpha) \tag{11}$$

$$\theta_i \sim G_0 \tag{12}$$

where the value of $V_i \prod_{j=1}^{i-1}(1 - V_j)$ is often called "stick-breaking weight", $0 < V_i \prod_{j=1}^{i-1}(1 - V_j) < 1$. At the step i, V_i is broken from the remainder of the unit-length stick, $1 \times \prod_{j=1}^{i-1}(1-V_j)$, $0 < V_i < 1$. This means that there are always some sticks of remaining length for the following atoms, $\sum_{i=1}^{\infty} V_i \prod_{j=1}^{i-1}(1-V_j) = 1$.$G_0$ is the base distribution, which is used to sample the atom θ_i from the stick-breaking weights $V_i \prod_{j=1}^{i-1}(1 - V_j)$, $1 \le \mathrm{i} < \infty$. The value of the kth element of π is:

$$\pi_k = \sum_{i=1}^{\infty} V_i \prod_{j=1}^{i-1} (1 - V_j) \prod (\theta_i = k) \tag{13}$$

where $\prod(\cdot)$ is the indicator function. There may be several stick-breaking weights given to the same atom π_k in π.

3. Latent Dirichlet allocation with incremental vocabulary

In ivLDA, we choose the uniform distribution as the G_0. As in our opinion, every word (the atom in DP) has the identical probability at the beginning. The probability of words in the topic k, depends on the document d only but not the base distribution G_0. It is stronger in theory, Ishwaran and Zarepour [21] indicated when the G_0 follows a uniform distribution, $G_0 \sim (\frac{1}{K}, \ldots, \frac{1}{K})$, and $\mathrm{K} \to \infty$, $\mathrm{P}(\theta_i = \theta_j | \mathrm{i} \neq \mathrm{j}) = 0$, which means that there is a one-to-one correspondence between $V_i \prod_{j=1}^{i-1}(1 - V_j)$ and π_k, $1 \le \mathrm{i},\ \mathrm{k} < \infty$, so we give every word one stick-breaking weight in ivLDA.

The base distribution G_0 of infvoc-LDA is similar to a N-gram model, far from the uniform distribution, and there may be several θ_i point to π_j, which can be seen in (13). But in infvoc-LDA, each atom is given only one stick-breaking weight, which it is not practical, will increase the computational complexity.

The generative process of ivLDA is the same as the generative of LDA, expect that LDA draw words from Dirichlet distribution, meanwhile, ivLDA draw words from Dirichlet process. ϕ(k) of ivLDA follows DP(β, G_0), and it corresponds to only one stick-breaking weight $V_w(k)$:

$$\mathrm{p}(V_w(\mathrm{k})|\beta) = \mathrm{BETA}(1, \beta) \tag{14}$$

and $\phi_w(k)$ follows the stick-breaking construction definition:

$$\phi_w(k) = V_w(k) \prod_{j=1}^{LOC(w,k)-1} \left(1 - V_{WORD(j,k)}(k)\right) \tag{15}$$

where LOC (w, k) is the function that returns the index of word w, WORD (I, k) is the function that returns the word of position i. We make these two functions to find the indices of words in the topic k distribution, because the indices are not fixed and they will be changed after the words are sorted. Based on DP, we obtain the likelihood function of w:

$$\begin{aligned}
&\mathrm{p}(\mathrm{w}|z, \beta) = \int p(w, V|z, \beta)\, dV = \int p(w|z, V)\, p(V|\beta)\, dV \\
&= \int \prod_k \prod_w \left\{ V_w(k) \prod_{j=1}^{LOC(w,k)-1} \left(1 - V_{WORD(j,k)}(k)\right) \right\}^{\hat{\phi}_w(k)} \\
&\qquad \times \prod_k \prod_w p(V_w(k)|\beta)\, dV \\
&= \int \prod_k \prod_w \frac{\Gamma(1+\beta)}{\Gamma(1)\Gamma(\beta)} V_w(k)^{\hat{\phi}_w(k)} \\
&\times (1 - V_w(k))^{\beta + \sum_{j=LOC(w,k)+1}^{\infty} \hat{\phi}_{WORD(j,k)}(k) - 1}\, dV \\
&= \int \prod_k \prod_w \frac{\Gamma(1+\beta)}{\Gamma(1)\Gamma(\beta)} \\
&\times \frac{\Gamma\left(1 + \hat{\phi}_w(k)\right) \Gamma\left(\beta + \sum_{j=LOC(w,k)+1}^{\infty} \hat{\phi}_{WORD(j,k)}(k)\right)}{\Gamma\left(1 + \beta + \sum_{j=LOC(w,k)}^{\infty} \hat{\phi}_{WORD(j,k)}(k)\right)}
\end{aligned} \tag{16}$$

where V is the set containing all the atoms in all topic distributions, $\hat{\phi}_w(k)$ is the sufficient statistics of $V_w(k)$. We obtain the likelihood function of z, which is the same as the BP's:

$$p(z|\alpha) = \prod_{j=1} \left\{ \frac{\Gamma(K\alpha)}{\Gamma\left(\sum_k \hat{\theta}_d(k)\right)} \prod_w \frac{\Gamma\left(\hat{\theta}_d(k) + \alpha\right)}{\Gamma(\alpha)} \right\} \tag{17}$$

Multiplying (16) by (17) we can obtain the joint probability of collapsed LDA, p(x, z). To maximize the likelihood of z, ivLDA also uses BP's approach to compute the posterior probability:

$$\begin{aligned}
&\mu_{w,d}(k) = \mathrm{p}\left(z^k_{w,d} = 1 | z^k_{-(w,d)}, \mathrm{x}\right) \\
&\propto \frac{p(w|z)}{p\left(w_{-(w,d)}|z_{-(w,d)}\right)} \frac{p(z)}{p\left(z_{-(w,d)}\right)} \\
&\propto \frac{\Gamma\left(1 + \hat{\phi}_w(k)\right) \prod_{i=1}^{LOC(w,k)-1} \Gamma\left(\beta + \sum_{j=i+1}^{\infty} \hat{\phi}_{WORD(j,k)}(k)\right)}{\prod_{i=1}^{LOC(w,k)} \Gamma\left(1 + \beta + \sum_{j=i}^{\infty} \hat{\phi}_{WORD(j,k)}(k)\right)} \\
&\times \frac{\prod_{i=1}^{LOC(w,k)} \Gamma\left(1 + \beta + \sum_{j=i}^{\infty} \hat{\phi}'_{WORD(j,k)}(k)\right)}{\Gamma\left(1 + \hat{\phi}'_w(k)\right) \prod_{i=1}^{LOC(w,k)-1} \Gamma\left(\beta + \sum_{j=i+1}^{\infty} \hat{\phi}'_{WORD(j,k)}(k)\right)} \\
&\times (\hat{\theta}_{-w,d}(k) + \alpha)
\end{aligned} \tag{18}$$

In (18), $\hat{\phi}'_i(k) = \hat{\phi}_i(k)$, i $\in$ vocabulary, except w, $\hat{\phi}'_w(k) = \hat{\phi}_{w,-d}(k)$, we can find that the only difference between $\Gamma\left(\beta + \sum_{j=i+1}^{\infty} \hat{\phi}_{WORD(j,k)}(k)\right)$, $1 \le \mathrm{i} \le \mathrm{LOC}(\mathrm{w}, \mathrm{k}) - 1$, and $\Gamma\left(1 + \beta + \sum_{j=i}^{\infty} \hat{\phi}_{WORD(j,k)}(k)\right)$, $2 \le \mathrm{i} \le \mathrm{LOC(w, k)}$ is the $\Gamma(\cdot)$ function. The values of β and $\sum_w \hat{\theta}_w(k)$ are often very large, so we ignore the difference and the (18) becomes:

$$\mu_{w,d}(k) \propto \frac{\left(1 + \hat{\phi}_{w,-d}(k)\right) \times \left(\hat{\theta}_{-w,d}(k) + \alpha\right)}{1 + \beta + \hat{\phi}_{-(w,d)}(k)} \tag{19}$$

where the sufficient statistics $\hat{\theta}_{-w,d}(k)$ and $\hat{\phi}_{w,-d}(k)$ are calculated by:

$$\hat{\theta}_{-w,d}(k) = \sum_{-w} x_{w,d}\mu_{w,d}(k) \tag{20}$$

$$\hat{\phi}_{w,-d}(k) = \sum_{-d} x_{w,d}\mu_{w,d}(k) \tag{21}$$

$$\hat{\phi}_{-(w,d)}(k) = \sum_{word,word\neq w} \hat{\phi}_{word}(k) + \hat{\phi}_{w,-d}(k) \tag{22}$$

The parameters, θ and ϕ, can be estimated by $\hat{\theta}$ and $\hat{\phi}$. θ can be estimated by Dirichlet normalization, which is the same as Eq. (5). We give the sufficient statistics $\hat{\phi}_w(k)$ to ϕ,

$$p(V_k(w)|\hat{\phi}_w(k), \beta) = BETA(1 + \hat{\phi}_w(k), \beta + \sum_{j=LOC(w,k)}^{\infty} \hat{\phi}'_{WORD(j,k)}(k)) \tag{23}$$

$$V_k(w) = \frac{1 + \hat{\phi}_w(k)}{1 + \beta + \sum_{j=LOC(w,k)}^{\infty} \hat{\phi}'_{WORD(j,k)}(k)} \tag{24}$$

where (24) is sampled from (23), and put (24) into (15), we obtain the normalized ϕ finally.

3.1. Truncation method

As mentioned above, DP is a distribution with an infinite dimension, but there is no need to make the vocabulary of LDA has infinite words. We just hope that the vocabulary contains all the words that appear in the current batch, because we do not need to process the words that do not appear in the current batch. So, we design a truncation method which can cut an infinite set of words into a finite subset. There are no words in the subset at the beginning, and then new words will be added into the subset from the batches. Sticks of unit length cannot be fully allocated, so we give the words, which have the highest indices in topic k distribution, $1 \leq k \leq K$, the probability:

$$\phi_w(k) = 1 - \sum_{word,word\neq w} \phi_{word}(k) \tag{25}$$

which ensures that $\sum_w \phi_w(k) = 1$.

At the beginning of the process, ivLDA finds the new words and add them into the vocabulary, after that it processes all the documents based on the new vocabulary. The only thing which is passed between batches is $\phi_{W\times K}$, where the W will increase when the new words are found.

However, the truncation method of infvoc-LDA only provides a truncation whose size is T_k, which can only contain the top T_k words, the remaining words will be lost. The words which want to enter the top T_k of the topic distribution must get enough sufficient statistics in the current batch. Some words that appear rarely in the current batch, have no chance to stay in the vocabulary. But in the following batches, they may carry more valuable information, and if the algorithm ignores them because they do not in the vocabulary will cause information loss. We know that our truncation method will make the topic distribution very large, but we do not think it is necessary to save the entire topic distribution into memory. We save it in the external storage, and move the needed parts into memory when using them. More solutions can be found in [12]. We know that it will increase the cost of computation, but the important words in batches will not be lost.

3.2. Sorting method

Though we can make the topic model have incremental vocabulary by Dirichlet Process, DP has some problem need to solve. First of all, the ordering of atoms in DP is very important. DP is a size biased distribution, words with lower indices. are likely to have higher probability than words with higher indices, therefore it is necessary to design a method to sort the atoms.

infvoc-LDA also take care of the atoms ordering problem, it also uses the sort method to sort vocabulary under every topic. But the way they sort is different from us, they use the word probability under their base distribution G0, multiply by the word sufficient statistics under topic distribution. The base distribution G_0 of infvoc-LDA is similar to a N-gram model, far from the uniform distribution, and there may be several θ_i point to π_j, but in infvoc-LDA, each atom is given only one stick-breaking weight, which it is not practical, will increase the computational complexity.

In the part above, we have proved that each word w in topic k corresponds to only one probability weight $V_i\prod_{j=1}^{i-1}(1-V_j)$. For the convenience of the management of words, we manually define the mapping between w and i. Since DP has biases of atoms locations [13,15], the order of atoms is very important. To sort the atoms, we provide two methods:

(1) We do not sort the words in each topic distribution, which means the words which have lower indices were found in the earlier batches. We named it ivLDA-Perp.
(2) We sort the words in every topic distribution, from largest to smallest depending on $\hat{\phi}_w(k)$. Due to the biases of atoms locations, the words which have high sufficient statistics have high probability in the topic distribution, and the words which have low sufficient statistics have low probability. The top words in the topic distribution can get more probability than others. The biases may have some negative effects on the accuracy of LDA. From the experiments, we find that this method does not perform well on perplexity. But on PMI, it has very high performance. The reason is that it focuses on the top words in the topic distribution, and there are always some words whose sufficient statistics $\hat{\phi}_w(k)$ are high enough. We named it ivLDA-PMI.

3.3. ivLDA-Perp

As mentioned above, ivLDA-Perp does not sort the words in each topic distribution. It is similar to OBP, but ivLDA-Perp can find new words and add them into the vocabulary which OBP cannot.

We found that the accuracy of LDA is related to the size of vocabulary W and the value of β, a larger vocabulary can make the accuracy of LDA higher. A reasonable explanation is that if we cannot make β in a sufficiently small range when W is large, $W\times\beta$ will be large enough to affect the topic–word distribution $\hat{\phi}(k)$, which will be very close to uniform distribution. On the contrary if we make the value of β too small, infinitely close to zero, $\hat{\phi}(k)$ will lose its smoothness. The value of W has no effect on the accuracy of ivLDA-Perp, since the update Eq. (19) of ivLDA is different from the update Eq. (3) of BP, the part $W\times\beta$ in the denominator of BP is replaced by $\beta+1$, and β is replaced by 1 in the nominator. We summarize ivLDA-Perp in **Algorithm 1**. The computation complexity of ivLDA-Perp is O(NNZ*K*T), NNZ is the numbers of the non-zero words, K is the topic size, T is the number of iterations.

Algorithm 1: ivLDA-Perp.

Require: $D_S, K, \alpha, \beta, x^s_{w,d}, T, S$

Ensure: $\phi_{K\times W}$

for s ← 1 to ∞ **do**
  load $x^s_{w,d}, d \in D_S$ in memory;
  find all the new words in D_S, then insert them into vocabulary;
  randomly give topics to the words in D_S, initialize the $\mu^k_{w,d}$;
  for t ← 1 to T **do**
    residual ← 0
    for d ∈ D_S,w ∈ d **do**
      for k ← 1 to K **do**
      // do not sort the words.
        $\hat{\phi}_w(k) \leftarrow \hat{\phi}_w(k) - x^s_{w,d} \cdot \mu_{w,d}(k)$;
        $\hat{\theta}_d(k) \leftarrow \hat{\theta}_d(k) - x^s_{w,d} \cdot \mu_{w,d}(k)$;
        $\mu^{new}_{w,d}(k) \leftarrow \frac{(\alpha+\hat{\theta}_d(k))\times(1+\hat{\phi}_w(k))}{1+\beta+\sum_{i\in vocabulary}\hat{\phi}_i(k)}$;
      end for
      for k ← 1 to K **do**
      // iterate on new vocabulary.
        $\mu_{w,d}(k) \leftarrow \frac{\mu^{new}_{w,d}(k)}{\sum_{k=1}^{K}\mu^{new}_{w,d}(k)}$
        residual ← residual + $x^s_{w,d} \times |\mu^{new}_{w,d}(k) - \mu_{w,d}(k)|$
        $\hat{\phi}_w(k) \leftarrow \hat{\phi}_w(k) + x^s_{w,d} \cdot \mu_{w,d}(k)$
        $\hat{\theta}_d(k) \leftarrow \hat{\theta}_d(k) + x^s_{w,d} \cdot \mu_{w,d}(k)$
      end for
    end for
    $\left(\frac{\text{residual}}{\sum_{w,d} x^s_{w,d}}\right) < 0.01$ then break;
  end for
end for

3.4. ivLDA-PMI

As is pointed out in [22,23], frequent words often belong to more than one topic, which makes the topics are similar to each other. This is called the Topic Duplication [22]. In [22], Y. Wang et al. provide a method that cluster the common words into one topic to avoid them dominating other topics by using asymmetric Dirichlet prior α_k instead of symmetric Dirichlet prior α. Common words will be clustered into the topic k when α_k is given a large value.

Comparing with the method above, ivLDA-PMI provides a new way to deal with the topic duplication. We multiply the message $\mu_{w,d}(k)$ by the index-wise coefficient $\prod_{j=1}^{LOC(w,k)-1}(1-\phi_{WORD(j,k)}(k))$, the update function of new message $\mu_{w,d}(k)$ becomes:

$$\mu_{w,d}(\mathrm{k}) = \frac{\left(1+\hat{\phi}_{w,-d}(k)\right)\times\left(\hat{\theta}_{-w,d}(k)+\alpha\right)}{1+\beta+\hat{\phi}_{-(w,d)}(k)} \times\left\{\prod_{j=1}^{LOC(w,k)-1}\left(1-\phi_{WORD(j,k)}(k)\right)\right\} \tag{26}$$

By adding the index-wise coefficient to the message update equation, $\mu_{w,d}(\mathrm{k})$ will be smaller when the word w has a higher probability in topic k, and in the topic k′, which is different from the topic k, the probability of the same word w will be lower and $\mu_{w,d}(\mathrm{k}')$ is larger. On the contrary, when the word w has a lower probability in topic k, $\mu_{w,d}(\mathrm{k})$ will become larger, the probability of the word w in the topic k′ will be higher and $\mu_{w,d}(\mathrm{k}')$ is smaller. $\mu_{w,d}(\mathrm{k})$ will be more stable and sparser with the increasing number of iterations.

The word which has a higher probability in a certain topic will have a lower probability in the other topics due to the index-wise coefficient, and each word will be assigned to a more specific topic. ivLDA-PMI sort the words in every topic from the largest to the smallest according to $\hat{\phi}_w(k)$ at the end of the processing of each batch, and a certain word will gradually concentrate on a certain topic k. We summarize ivLDA-PMI in **Algorithm 2**. The computation complexity of ivLDA-PMI is O(NNZ*K*T), NNZ is the numbers of the non-zero words, K is the topic size, T is the number of iterations.

Table 1
Summarization for datasets.

Data sets	D_train	D_test	W
ENRON	37 874	1987	28 102
WIKI	18 606	2152	83 470
NIPS	1 350	150	12 419
20NEWS	1 900	100	36 863
NYTIMES	274 771	4981	102 660

Algorithm 2: ivLDA-PMI.

Require: $D_S, K, \alpha, \beta, x^s_{w,d}, T, S$

Ensure: $\phi_{K\times W}$

for s ← 1 to ∞ **do**
  load $x^s_{w,d}, d \in D_S$ in memory;
  find all the new words in D_S, then insert them into vocabulary;
  randomly give topics to the words in D_S, initialize the $\mu^k_{w,d}$;
  for t ← 1 to T **do**
    residual ← 0
    for d ∈ D_S,w ∈ d **do**
      for k ← 1 to K **do**
      // sort the words.
        $\varphi_w(k) \leftarrow \prod_{j=1}^{LOC(w,k)-1}(1-\frac{1+\hat{\phi}_w(k)}{1+\beta+\sum_{i\in vocabulary}\hat{\phi}_i(k)})$;
        $\hat{\phi}_w(k) \leftarrow \hat{\phi}_w(k) - x^s_{w,d} \cdot \mu_{w,d}(k)$;
        $\hat{\theta}_d(k) \leftarrow \hat{\theta}_d(k) - x^s_{w,d} \cdot \mu_{w,d}(k)$;
        $\mu^{new}_{w,d}(k) \leftarrow \frac{\varphi_w(k)\times(\alpha+\hat{\theta}_d(k))\times(1+\hat{\phi}_w(k))}{1+\beta+\sum_{i\in vocabulary}\hat{\phi}_i(k)}$;
      end for
      for k ← 1 to K **do**
      // iterate on new vocabulary
        $\mu_{w,d}(k) \leftarrow \frac{\mu^{new}_{w,d}(k)}{\sum_{k=1}^{K}\mu^{new}_{w,d}(k)}$
        residual ← residual + $x^s_{w,d} \times |\mu^{new}_{w,d}(k) - \mu_{w,d}(k)|$
        $\hat{\phi}_w(k) \leftarrow \hat{\phi}_w(k) + x^s_{w,d} \cdot \mu_{w,d}(k)$
        $\hat{\theta}_d(k) \leftarrow \hat{\theta}_d(k) + x^s_{w,d} \cdot \mu_{w,d}(k)$
      end for
    end for
    $\left(\frac{\text{residual}}{\sum_{w,d} x^s_{w,d}}\right) < 0.01$ then break;
  end for
  sort words in every topic, depends on $\hat{\phi}_w(k)$, from large to small.
end for

4. Experiments

In this section we evaluate the performance of our models: ivLDA-Perp and ivLDA-PMI with five publicly available datasets: NIPS, WIKI, ENRON, 20NEWS, NYTIMES. Details of the datasets are shown in Table 1, where the D_train is the number of documents in train sets, D_test is the number of documents in test sets and W is the size of corpus.

We compare our models: ivLDA-Perp and ivLDA-PMI with Infvoc-LDA and the other three new online LDA algorithms with fixed vocabulary: OBP, OGS, and OVB.

We use the Grid Search method which KYM Chen and Y Wang proved which is a useful tool in [24] to find the best parameters of each model. For OVB and infvoc-LDA, the parameter $\tau_0 = 1024$ and $\kappa = 0.6$. For Infvoc-LDA, we give the size of Truncation Ordered Set (TOS) W. For All the algorithms, the hyperparameters $\alpha = 0.01$. However, the choice of β is somewhat different. Because stick-breaking process has the biases of atoms locations, if the hyperparameter β is given a small value, the words which have low indices in topic k distribution will occupy most of the probability. So, for ivLDA and Infvoc-LDA, we make $\beta = 1000$. But for other online LDA algorithms with fixed vocabulary, we give $\beta = 0.01$ as recommended in [10,12,16]. For all online

LDA algorithms with fixed vocabulary, we use the vocabulary of corpus as the LDA's vocabulary, to compare our models with the best performing online LDA algorithms with fixed vocabulary.

4.1. Performance measure

We use three measures to examine all the algorithms:

(1) Predictive Perplexity: We use predictive perplexity, which has been widely used in previous work [25,26], as a performance measure to examine the test sets. Perplexity is the inverse of the average likelihood of each word with the parameters θ and ϕ of LDA. Lower value of perplexity indicates better generalization performance. The perplexity is calculated as:

$$\text{Perp} = \exp\left\{-\frac{\sum_{w,d} x_{w,d} \log\left(\sum_k \theta_d(k)\,\phi_w(k)\right)}{\sum_{w,d} x_{w,d}}\right\} \tag{27}$$

In this paper, the perplexity is computed as follows: First of all, we run the algorithms on the train sets and fixed ϕ which is obtained from the train sets. Then, we run the algorithms on the 80% words of each document in the test sets to get θ_d of every document d. Finally, we compute the perplexity on the rest 20% words of each document in the test sets using Eq. (27). Words are selected randomly in the test sets.

(2) Point-wise Mutual Information (PMI) [27]: Each topic is shown as a list of the high-frequency words (for example, top ten words in each topic) in LDA. PMI is an indicator of the degree of coherence of the topic, and it will be high when the algorithm produces highly coherent topic distributions. PMI is computed as:

$$\text{PMI} = \sum_{i=2}^{M}\sum_{j=1}^{i-1} \log\left\{\frac{Q\left(w_i^k, w_j^k\right)+\epsilon}{Q\left(w_i^k\right)Q\left(w_j^k\right)}\right\} \tag{28}$$

where $W^k = (w_1^k, \ldots, w_M^k)$ is the list of the most likely M words in topic k, ϵ is a small constant used to avoid the fraction of the log function being equal to zero. Q(w) is the number of documents which contain the word w. $Q(w_1, w_2)$ is the number of documents containing at least a pair of words w_1 and w_2.

(3) Documents Classification: LDA represents the document d as a vector of the document–topic distribution θ_d, so we can evaluate LDA algorithms through the performance of representation. We use θ_d as the feature of document d, and use the category of the documents as the labels of classification. We use 80% of the documents in corpus as the train sets, and use the remaining documents as test sets. We use 5-fold cross-validation method. We use the linear SVM and Random Forest classifier for document classification. The higher the average classification accuracy of the test sets, the better the documents representation ability of the algorithm.

4.2. Performance on predictive perplexity

All online LDA algorithms are given the batch size D_s as one of the parameters. According to the previous studies, the batch size D_s has a slight effect on the perplexity of LDA [12,26]. Therefore, we evaluate the algorithms with different D_s. In our experiments, we check the perplexity on the batches of size $D_s = 256, 512, 1024, 2048, 4096$ from three large corpora: ENRON, NYTIMES, and WIKI. For the small corpus NIPS, we check the perplexity on the batches of size $D_s = \{100, 200, 300, 400, 500\}$. Fig. 1 shows the predictive perplexity of each algorithm with different D_s. We can find that, as D_s increases, the perplexity of OBP, OGS, and ivLDA-Perp will decrease slightly, because the online algorithms are more similar to the batch algorithms when D_s becomes larger. However, OVB and infvoc-LDA tend to perform worse when D_s increases, this may be attributed to the approximate objective functions of OVB and infvoc-LDA. The performance of ivLDA-PMI is not stable when D_s increases in all corpora due to the index-wise coefficient in Eq. (26) which is used to make sure that most of the probability of the topic is given to the words with low indices, and the words with high indices may not get enough probability. In all cases, ivLDA-Perp achieves the best performance in all corpora, showing the highest accuracy.

The topic size k is another important parameter of LDA. Fig. 2 shows the predictive perplexity with different k values, where $k = \{100, 200, 300, 400, 500\}$. Since a smaller batch will take up more memory space, we choose $D_s = 1024$ for large corpus and $D_s = 300$ for small corpus. ivLDA-Perp still performs best in all corpora. OGS and OBP have higher perplexity than ivLDA-Perp. OVB have higher perplexity than OGS, OBP, and ivLDA-Perp. infvoc-LDA still shows the worst performance. And the performance of ivLDA-PMI is still unstable when k increases.

Fig. 3 shows the training time of the algorithms with different topic sizes. We can find that ivLDA-Perp and OBP have similar training time in the four corpora. OGS is slower than OBP and ivLDA-Perp, because it is an MCMC algorithm and it needs more iterations than OBP and ivLDA-Perp. OVB is much slower, because $p(z_{d,w})$ of OVB contains the digamma function [8], which is a very time-consuming function. infvoc-LDA and ivLDA-PMI are the slowest algorithms, both of them need to spend a lot of time sorting the words.

Fig. 4 shows the predictive perplexity of vocabulary of different sizes. The *x*-axis indicates the size of vocabulary. When x is less than 1, we extract the words which are the top x% words from the vocabulary of corpus to organize the new vocabulary. When x is greater than 1, we add the $(1 - x)\%$ new words to the vocabulary of corpus, instead of the documents, and then we use the new vocabulary to compute the perplexity of algorithms. Since ivLDA-PMI and infvoc-LDA do not perform well on perplexity, they may not be suitable for the evaluation on perplexity. So, we just compare ivLDA-Perp with other online LDA algorithms with fixed vocabulary. From the experiments, we can clearly find that as the size of vocabulary increases, the perplexity of all algorithms increases, but ivLDA-Perp is more stable than the others. Online LDA algorithms with fixed vocabulary perform worse as the size of vocabulary increases. Fig. 4 confirms that ivLDA-Perp performs best when process the new words appear in streams.

4.3. Performance on PMI

Fig. 5 shows the values of PMI of all algorithms, when $K = 100$, $\epsilon = 1000$, $M = 10$. We examine the top 10 words in every topic using Eq. (28) and use the average value as the algorithm's PMI. We add a separate corpus to make the size of $Q(\cdot)$ bigger, because we think that making the size of $Q(\cdot)$ bigger can make the PMI higher. We make the size of the TOS of infvoc-LDA the same as the size of vocabulary of corpus. Fig. 5 shows that ivLDA-PMI achieves the best performance on the PMI. This verifies that sorting the words of every topic is a good way to find the most valuable words in every topic. infvoc-LDA has a better performance than the other algorithms expect ivLDA-PMI. Other algorithms perform worse than ivLDA-PMI and infvoc-LDA when focusing on the top words in the topic.

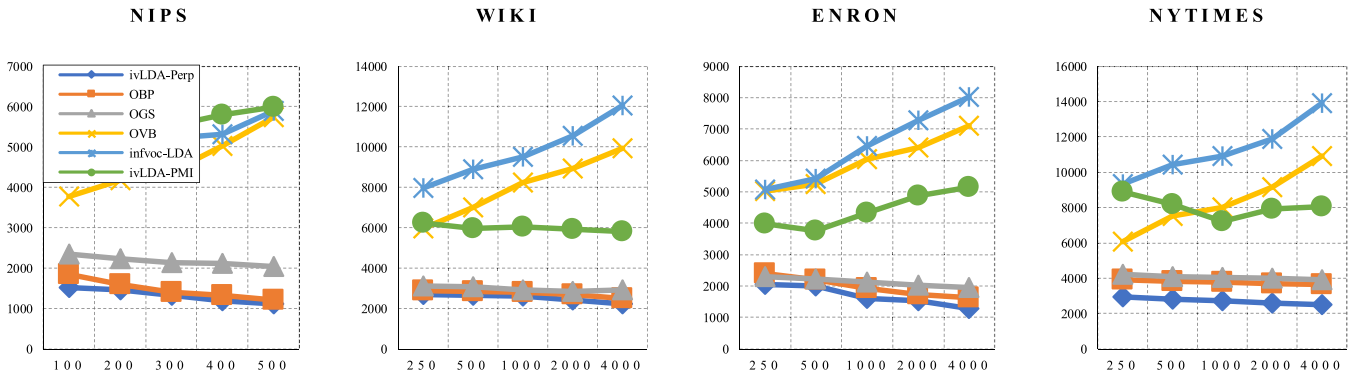

Fig. 1. Predictive perplexity as a function of batch size when $K = 100$ on NIPS, $K = 300$ on WIKI, ENRON, and NYTIMES.

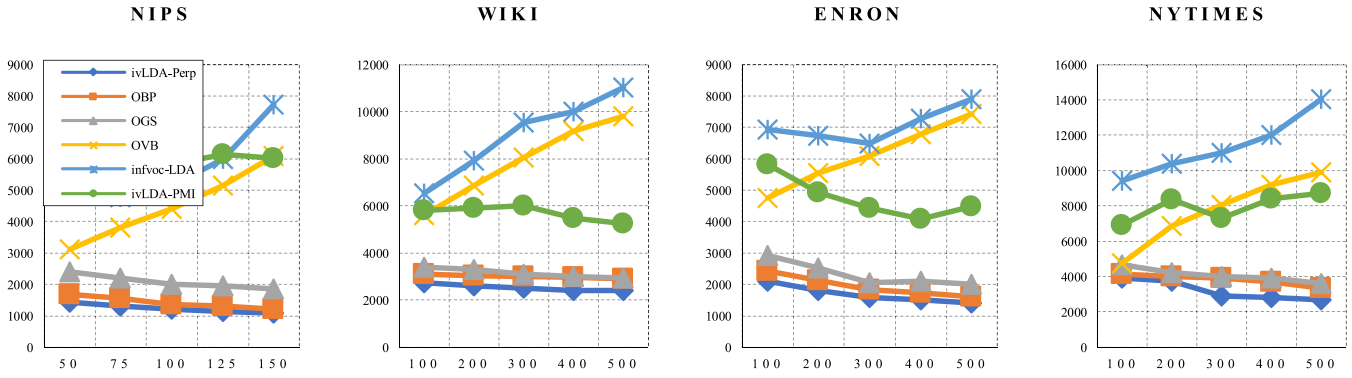

Fig. 2. Predictive perplexity as a function of the number of topics K when $K = 300$ on NIPS, $K = 1024$ on WIKI, ENRON, and, NYTIMES.

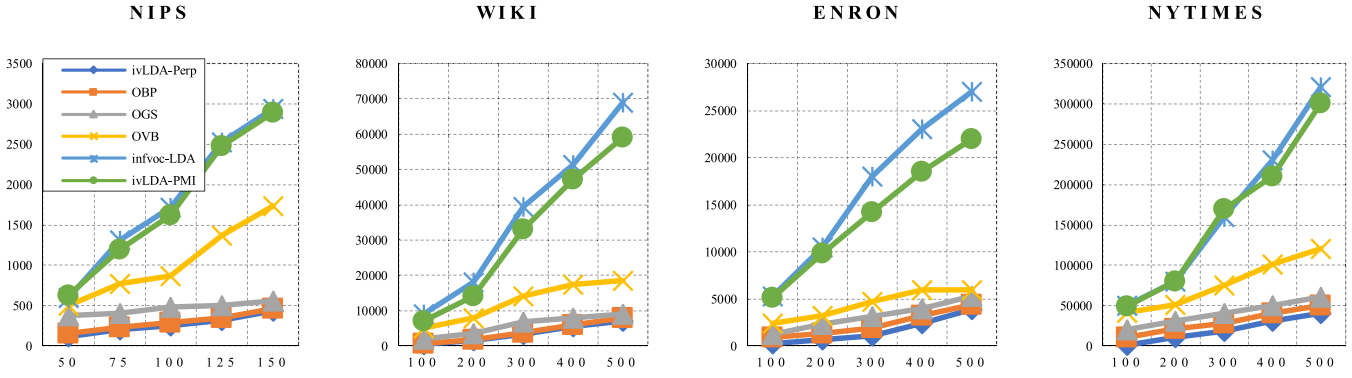

Fig. 3. Training time(s) as a function of the number of topics K when $Ds = 300$ on NIPS, $Ds = 1024$ on WIKI, ENRON, and NYTIMES.

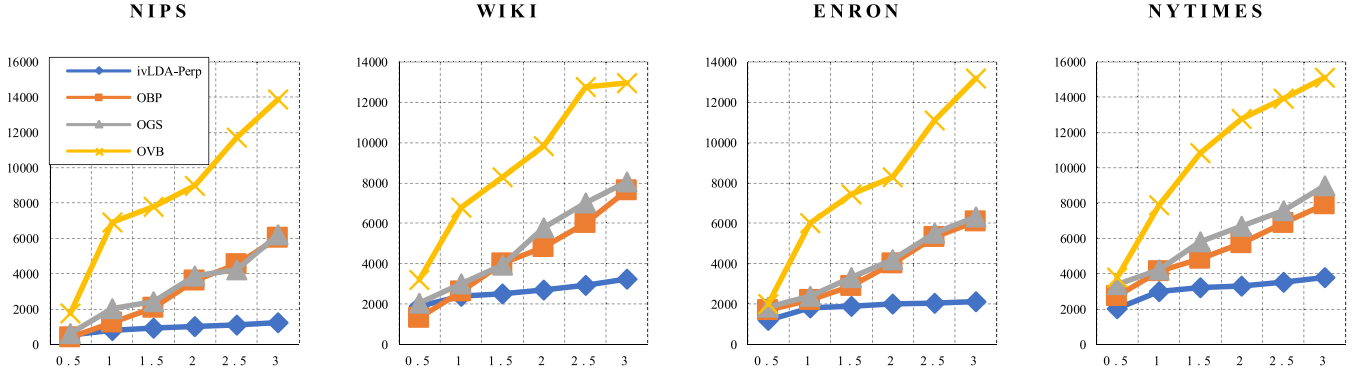

Fig. 4. Predictive perplexity as a function of size of vocabulary when $K = 300$, $Ds = 300$ on NIPS and $Ds = 1024$ on WIKI, ENRON, and NYTIMES.

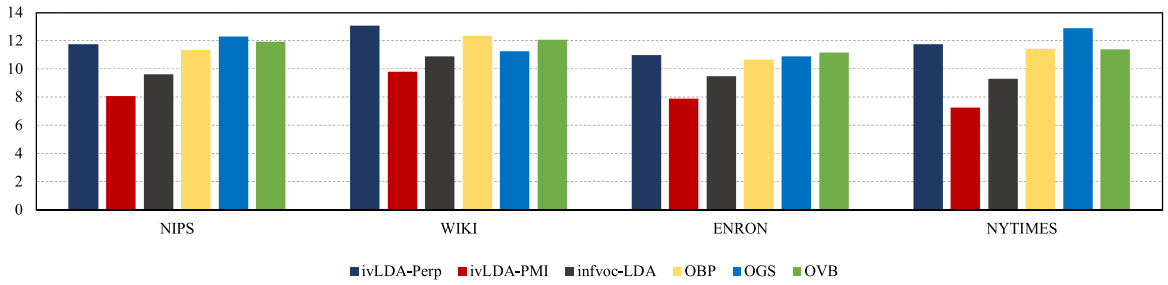

Fig. 5. The reciprocals of PMI of each algorithm when $K = 100$ and $Ds = 300$ on NIPS, $Ds = 1024$ on WIKI, ENRON, and NYTIMES. The lower the function, the better the performance.

Table 2
Document classification performance use SVM on CORA and 20NEWS.

Datasets	CORA			20NEWS		
K	*50*	*200*	*500*	*50*	*200*	*500*
ivLDA-Perp	0.722	0.76325	0.7511	0.6312	0.6512	0.6499
ivLDA-PMI	**0.7841**	**0.83021**	**0.8102**	**0.7413**	**0.7913**	0.7713
infvoc-LDA	0.707	0.8162	0.8011	0.7013	0.7900	**0.7815**
OBP	0.7125	0.7533	0.7344	0.6294	0.6670	0.6402
OGS	0.6987	0.7652	0.7121	0.6451	0.6825	0.6801
OVB	0.7021	0.7382	0.6912	0.6315	0.6723	0.6511

Table 3
Document classification performance use random forest on CORA and 20NEWS.

Datasets	CORA			20NEWS		
K	*50*	*200*	*500*	*50*	*200*	*500*
ivLDA-Perp	0.7314	0.7706	0.7586	0.6415	0.676	0.6614
ivLDA-PMI	**0.7883**	**0.8411**	**0.8174**	**0.7536**	0.8046	**0.7843**
infvoc-LDA	0.7211	0.8320	0.8062	0.7323	**0.8091**	0.7814
OBP	0.7234	0.751	0.7414	0.6476	0.6888	0.6552
OGS	0.7076	0.76325	0.7212	0.6577	0.6973	0.6837
OVB	0.7041	0.7595	0.7094	0.6455	0.6946	0.661

4.4. Performance on documents classification

Tables 2 and 3 shows the accuracy of documents classification using the feature θ. ivLDA-PMI achieves the highest accuracy on NIPS and 20NEWS, except on 20NEWS when k $=$ 500 using SVM and when $k = 200$ using Random Forest. infvoc-LDA has also yielded good results at some points which are very close to ivLDA-PMI. ivLDA-Perp is better than the LDA algorithms with fixed vocabulary but worse than ivLDA-PMI and infvoc-LDA on document classification. From the experiments, we can also confirm that focusing on the topic-representation words can make the θ more abundant which can make the performance of documents classification better. Furthermore, we also find that the accuracy increases when the size of topics k is from 50 to 200, but it decreases when k is from 200 to 500. A possible reason is that the size of topics k in p(w|k) does not fit the corpus, which may lead to a lower probability of w, so that the algorithms cannot represent the effective information in the corpus well.

5. Conclusion

To address the drawbacks of LDA algorithms with a fixed vocabulary, ivLDA with topic–word distribution resulting from Dirichlet progress that has infinite atoms instead of Dirichlet distribution is proposed. ivLDA has an incremental vocabulary where new words from batches can be added that enables the topic models to process data streams. In addition, two methods to manage the indices of the words are presented, namely ivLDA-Perp and ivLDA-PMI. A comparison is performed of them with infvoc-LDA and other state-of-the-art algorithms with fixed vocabulary through experiments, which reveals that ivLDA-Perp is capable to achieve high accuracy and ivLDA-PMI can identify the most useful words to represent the topic.

Our models can be applied to extracting the topic model of the real time data stream, like the analysis of traffic data stream. Despite this, there remains a lot to do. In the future, we will extent ivLDA-PMI to make it more practical and find ways to expedite it.

Acknowledgment

This paper is supported by the National Natural Science Foundation of China (61572339).

References

[1] D.M. Blei, A.Y. Ng, M.I. Jordan, Latent Dirichlet allocation, J. Mach. Learn. Res. 3 (2003) 993–1022.
[2] H. Jelodar, Y. Wang, C. Yuan, X. Feng, Latent Dirichlet allocation (LDA) and topic modeling: models, applications, a survey, 2017, arXiv preprint arXiv:1711.04305.
[3] Y.K. Tang, X.L. Mao, H. Huang, Labeled phrase latent Dirichlet allocation and its online learning algorithm, Data Min. Knowl. Discov. (2018) 1–28.
[4] T.L. Griffiths, M. Steyvers, Finding scientific topics, Proc. Natl. Acad. Sci. 101 (2004) 5228–5235.
[5] J. Zeng, W.K. Cheung, J. Liu, Learning topic models by belief propagation, IEEE Trans. Pattern Anal. Mach. Intell. 35 (5) (2013) 1121–1134.
[6] J. Lehečka, A. Pražák, Online LDA-based language model adaptation, in: International Workshop on Temporal, Spatial, and Spatio-Temporal Data Mining, Springer, Cham, 2018, pp. 334–341.
[7] D. Blei, Scalable Topic Modeling: Online Learning, Diagnostics, and Recommendation, Columbia Univ., New York, New York, United States, 2017.
[8] a.F.M. Hoffman, D. Blei, Online inference of topics with Latent Dirichlet Allocation, in: NIPS, 2010, pp. 856–864.
[9] J. Lehečka, A. Pražák, Online LDA-based language model adaptation, in: International Workshop on Temporal, Spatial, and Spatio-Temporal Data Mining, Springer, Cham, 2018, pp. 334–341.
[10] D.L. Yao, T.L. Griffths, Efficient methods for topic model inference on streaming document collections, in: KDD, 2009, pp. 937–946.
[11] Y. Papanikolaou, J.R. Foulds, T.N. Rubin, G. Tsoumakas, Dense distributions from sparse samples: improved Gibbs sampling parameter estimators for LDA, J. Mach. Learn. Res. 18 (1) (2017) 2058–2115.
[12] J. Zeng, Z.-Q. Liu, X.-Q. Cao, Online belief propagation for topic modeling, 2012, arXiv:1210.2179 [cs.LG].
[13] J. Sethuraman, A constructive definition of Dirichlet priors, Statist. Sinica 4 (1994) 639–650.
[14] J. Zhang, J. Zeng, M. Yuan, W. Rao, J. Yan, LDA revisited: Entropy, prior and convergence, in: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, ACM, 2016, pp. 1763–1772.
[15] K. Zhai, J. Boyd-Graber, Online Latent Dirichlet Allocation with infinite vocabulary, in: ICML, 2013, pp. 561–569.
[16] a.D. Mimmo David, Hoffman Matthew, Sparse stochastic inference for Latent Dirichlet Allocation, in: ICML, 2012.
[17] J. Zeng, X.-Q. Cao, Z.-Q. Liu, Residual belief propagation for topic modeling, in: ADMA, 2012, pp. 739–752.
[18] J. Zeng, Z.-Q. Liu, X.-Q. Cao, A new approach to speeding up topic modeling, 2012, arXiv:1204.0170 [cs.LG].
[19] T. Ferguson, A Bayesian analysis of some nonparametric problems, Ann. Statist. 1 (2) (1973) 209–230.
[20] T. Griffiths, Z. Ghahramani, Infinite Latent Feature Models and the Indian Buffet Process, Gatsby Unit Technical Report GCNU-TR-2005-001, 2005.

[21] H. Ishwaran, M. Zarepour, Dirichlet prior sieves in finite normal mixtures, Statist. Sinica (2002) 941–963.
[22] Y. Wang, X. Zhao, Z. Sun, et al., Towards topic modeling for big data[J]. arXiv preprint arXiv:1405.4402, 2014.
[23] D. Wallach, H. Mimno, Rethinking LDA: Why priors matter, in: NIPS, 2009, pp. 1973–1981.
[24] K.Y.M. Chen, Y. Wang, Latent Dirichlet Allocation, Technical report, Department of Electrical and Computing Engineering, University of California, San Diego 4, 2011, pp. 87–110.
[25] P.A. Asuncion, M. Welling, Y.W. Teh, On smoothing and inference for topic models, in: UAI, 2009, pp. 27–34.
[26] L. Huang, J. Ma, C. Chen, Topic detection from microblogs using T-LDA and perplexity, in: 2017 24th Asia-Pacific Software Engineering Conference Workshops, APSECW, IEEE, 2017, pp. 71–77.
[27] S.K.D. Newman, L. Cavedon, External evaluation of topic models, in: Australasian Document Computing Symposium, 2012, pp. 11–18.