X-DMM: Fast and Scalable Model Based Text Clustering

Linwei Li,[1] **Liangchen Guo,**[1] **Zhenying He,**[1,2,3] **Yinan Jing,**[1,2,3] **X. Sean Wang**[1,2,3]
[1]School of Computer Science and Technology, Fudan University
[2]Shanghai Key Lab of Data Science
[3]Shanghai Institute of Intelligent Electronics & Systems, China
{lwli15, lcguo17, zhenying, jingyn, xywangCS}@fudan.edu.cn

Abstract

Text clustering is a widely studied problem in the text mining domain. The Dirichlet Multinomial Mixture (DMM) model based clustering algorithms have shown good performance to cope with high dimensional sparse text data, obtaining reasonable results in both clustering accuracy and computational efficiency. However, the time complexity of DMM model training is proportional to the average document length and the number of clusters, making it inefficient for scaling up to long text and large corpora, which is common in real-world applications such as documents organization, retrieval and recommendation. In this paper, we leverage a symmetric prior setting for Dirichlet distribution, and build indices to decrease the time complexity of the sampling-based training for DMM from $O(K * L)$ to $O(K * U)$, where K is the number of clusters, L the average length of document, and U the average number of unique words in each document. We introduce a Metropolis-Hastings sampling algorithm, which further reduces the sampling time complexity from $O(K*U)$ to $O(U)$ in the nearly-to-convergence training stages. Moreover, we also parallelize the DMM model training to obtain a further acceleration by using an uncollapsed Gibbs sampler. We combine all these optimizations into a highly efficient implementation, called X-DMM, which enables the DMM model to scale up for long and large-scale text clustering. We evaluate the performance of X-DMM on several real world datasets, and the experimental results show that X-DMM achieves substantial speed up compared with existing state-of-the-art algorithms without clustering accuracy degradation.

Introduction

Text clustering is a popular problem in the field of data mining. The text data has the property of high dimensionality and sparsity (there are hundreds of thousands of word tokens in the lexicon, but only hundreds of words in each document). For such cases, the accuracy and computational efficiency of similarity-based clustering algorithms are always far from satisfactory. The model-based clustering algorithms have shown better performance to cope with high dimensional text data. In (Yin and Wang 2014), researchers proposed a Gibbs Sampling algorithm for the Dirichlet Multinomial Mixture model (GSDMM) to deal with the text clustering task. GSDMM assumes that the words in each document are generated by a single mixture component (i.e. a cluster), and uses a collapsed Gibbs sampling algorithm for model training. GSDMM achieves good balance in both completeness and homogeneity of the clustering results. It can process high dimensional sparse text data efficiently in terms of both computation and memory resources, and converge quickly. On real world datasets, GSDMM is proved to deliver the best clustering performance to date.

Although GSDMM uses bag-of-words model to represent each document and the order of words is ignored, the time complexity of GSDMM, as discussed in (Yin and Wang 2014), is still proportional to the length of the document. In this paper, we leverage the symmetric prior for the Dirichlet distribution to optimize the sequential multiplication by constructing indices. This reduces the time complexity of sampling each document from $O(L)$ to $O(U)$ (here L is the length of the document, and U is the number of unique words in the document). With this optimization, the time used for sampling is significantly reduced, especially for long text datasets.

In GSDMM, the complexity of sampling each document is also proportional to the number of clusters K. For large-scale document datasets, the number of clusters may be very large (e.g., $10^3 - 10^4$), which leads to expensive computational cost for each sampling. FGSDMM+(Yin and others 2016) adopts an online initialization scheme for clustering, and reduces the time complexity of clustering to be proportional to the number of non-empty clusters. In this work, we observe that the word distribution in the clusters varies slowly between each iteration, especially in the nearly-to-convergence stages. Thus we introduce a Metropolis-Hastings sampling method to reduce the time of sampling each document from $O(K * U)$ to $O(U)$ in each iteration, which can significantly reduce the sampling time for large text datasets.

For large-scale text datasets, we need resort to parallelization for acceleration. However, unlike its competitor similarity-based approaches such as Spherical K-means (Dhillon and Modha 2001) and K-medoids (Park and Jun 2009) that are embarrassingly parallelizable, the sampling process of GSDMM is strictly sequential and can not be parallelized. In this work, we introduce an uncollapsed Gibbs sampler to parallelize the model training, and show its scal-

ability on real world datasets.

The main contributions of this paper are summarized as follows:

1. We leverage a symmetric setting for Dirichlet prior and construct indices to optimize the sequential multiplication in the sampling. Thus the time complexity of sampling each document is reduced from $O(K * L)$ (proportional to the length of the document) to $O(K * U)$ (proportional to the number of unique words in the document).
2. We introduce a Metropolis-Hastings sampling algorithm to further reduce the time complexity of sampling each document from $O(K * U)$ to $O(U)$ in the nearly-to-convergence training stages.
3. We present an uncollapsed Gibbs sampler to parallelize the training of DMM model. For large datasets, there are enough parallelisms for model training. This leads to an improvement in scalability which is nearly linear to the number of processors.
4. We combine all of the above designs into an end-to-end efficient implementation, X-DMM, and present extensive evaluations of X-DMM on real-world datasets. The experimental results show that X-DMM substantially outperforms the existing state-of-the-art methods in efficiency without clustering accuracy degradation, especially for long and large text datasets.

The remainder of this paper begins with a review of relevant literature, and then briefly reviews GSDMM. In Section X-DMM we introduce the optimizations adopted by X-DMM in detail, including experimental results regarding each individual optimization. Finally we demonstrate the end-to-end experimental results in Section Experiments.

Related Works

General surveys and experimental comparisons about text clustering algorithms can be found in (Aggarwal and Zhai 2012; Anastasiu, Tagarelli, and Karypis 2013) and (Zhao and Karypis 2004). Roughly speaking, text clustering methods can be categorized into two categories: similarity-based methods and model-based methods.

Similarity-based methods use vector space model to represent each document as a vector point, and then use some similarity metrics to cluster the points, including partition-based algorithms (Dhillon and Modha 2001; Park and Jun 2009), density-based algorithms (Jain 2010), and hierarchical algorithms (Zhao, Karypis, and Fayyad 2005). Similarity-based methods are usually easy to implement. However, the clustering accuracy of similarity-based methods are generally low, because the similarity/distance metrics is more or less meaningless for high-dimensional data like text.

Model-based methods assume the data points (i.e. documents) are generated from a mixture model, and then use some inference methods such as EM and Gibbs sampling to estimate the model parameters. The Gaussian Mixture Model (GMM) assumes the data points are generated from a mixture of Gaussian distributions, and is the most widely used mixture model (Christopher 2016). However, the training complexity of GMM is too large for high dimensional text data. In EDCM (Elkan 2006), DPMFS (Yu, Huang, and Wang 2010) and DMAFP (Huang et al. 2013), Dirichlet distributions are adopted to model the text clustering, and have shown better performance in clustering accuracy. However, these models are more or less complex and slow to convergence.

In (Nigam et al. 2000), a Dirichlet Multinomial Mixture (DMM) model was introduced for both text classification and clustering, and an EM-based algorithm was used to estimate the model parameters. In (Yin and Wang 2014), researchers proposed a Gibbs sampling algorithm for the DMM model (GSDMM), obtaining the best clustering performance to date on real-world datasets. FGSDMM+ (Yin and others 2016) improved the training algorithm of GSDMM by adopting an online initialization scheme, reducing the training time from proportional to the number of clusters to proportional to the number of non-empty clusters. However, the training time of GSDMM and FGSDMM+ are still proportional to the average length of the documents and the number of clusters. Moreover, the algorithm is strictly sequential and can not be processed in parallel, thus can not scale up for long text and large-scale datasets. In this paper, we work on these problems and substantially improve the efficiency without losing the clustering accuracy.

Topic models such as LDA (Blei, Ng, and Jordan 2003), LSI (Papadimitriou et al. 2000) and pLSA (Hofmann 1999) assume that the words in a document are generated by first choosing a latent topic for each word, and then using the topic to generate the word itself. Hierarchical topic models (Blei, Griffiths, and Jordan 2010; Paisley et al. 2015) extended the topic model to more complex hierarchical structures. There are also works (Newman et al. 2009; Li et al. 2014) aiming to optimize and parallelize the topic model training. In (Yin and Wang 2014; Lu, Mei, and Zhai 2011) researchers investigated the performance of topic models on text clustering. Because the topic model assumes a latent variable for every word in each document, which does not fit the demand for clustering, the clustering accuracy of this method is poorer than that of DMM. Also, compared with DMM, the large number of latent variables (one latent variable per word) requires much more training time for those complex topic models.

Preliminaries

The DMM Model

The Dirichlet Multinomial Mixture (DMM) model (Nigam et al. 2000) is a probabilistic generative model for documents. It assumes that there are K mixture components (K clusters) in the document set, and each mixture component, which corresponds to a cluster, is a multinomial distribution over vocabulary. To generate a document, the DMM model assumes that a mixture component should be selected from K mixture components, and then the words in the documents are drawn from the multinomial distribution of that mixture component. Formally, the generative process of the model

Table 1: Notations for DMM model

Notation	Description
D	Number of documents in corpus
T	Number of distinct words in vocabulary
K	Number of clusters
α, β	Prior parameters for Dirichlet distribution
$\overrightarrow{d_i}$	The ith document
z_i	Cluster assignment of ith doc
n_{kt}	Count of word t in kth cluster
n_k	Count of words in kth cluster
N_{it}	Count of word t in ith document
N_i	Count of words in ith document
m_k	Count of documents in kth cluster
$\vec{\varphi_k}$	Word distribution of mixture k
φ_{kt}	Proportion of word t in the mixture k
θ	Distribution over mixture components

works as follows:

1. For each of the K mixture components (clusters) in the document set, draw $\overrightarrow{\varphi_k}$ from a Dirichlet prior, $\overrightarrow{\varphi_k} \sim Dirichlet(\alpha)$
2. Draw topic proportion $\overrightarrow{\theta}$ in the document set, $\overrightarrow{\theta} \sim Dirichlet(\beta)$
3. For each of the M documents $\overrightarrow{d_i}$:
   (a) Choose a mixture component (cluster) $z_i \sim multinomial(\overrightarrow{\theta})$.
   (b) For each word w in snippet $\overrightarrow{d_i}$:
   Choose $w \sim p(w|z_i, \overrightarrow{\varphi_k})$, a multinomial probability conditioned on the mixture component z_i.

For ease of reading, Table 1 lists the notation used in DMM model.

Training algorithm of DMM

As a probabilistic generative model, the training algorithm of the DMM model can be either an EM algorithm or a Markov Chain Monte Carlo (MCMC) algorithm. GSDMM introduced a collapsed Gibbs sampling (a MCMC method) algorithm for the DMM training. GSDMM can infer the number of clusters automatically, and is fast to converge. Algorithm 1 demonstrates the details of GSDMM:

Algorithm 1 The GSDMM algorithm

1: Initialize: for each document $\overrightarrow{d_i}$ in the corpus, randomly assign a cluster number z_i
2: Update the count statistics of the documents and the words in each cluster, n_{kt}, n_k, and m_k
3: Re-scan the documents and use collapsed Gibbs sampling to re-sample each document's cluster assignment $z_i \sim p(z_i = k|\overrightarrow{\mathbf{z}}_{\neg i}, \overrightarrow{\mathbf{d}}, \alpha, \beta)$
4: Repeat Step 3 until convergence
5: Output the cluster assignments of all the documents

The derivation of conditional probability distribution $p(z_i = k|\overrightarrow{\mathbf{z}}_{\neg i}, \overrightarrow{\mathbf{d}}, \alpha, \beta)$ was given in (Yin and Wang 2014), and the final form of $p(z_i)$ is as follows:

$$p(z_i = k|\overrightarrow{\mathbf{z}}_{\neg i}) \propto (m_{k,\neg i} + \alpha_k) \cdot \frac{\prod_{w \in \overrightarrow{d_i}} (\prod_{j=1}^{N_{iw}} (n_{kw,\neg i} + \beta_w + j - 1))}{\prod_{j=1}^{N_i} (n_{k,\neg i} + \sum_{t=1}^{T} \beta_t + j - 1)} \quad (1)$$

In FGSDMM+ (Yin and others 2016), an online clustering scheme is adopted in the initialization phase of model training, which can reduce the training time when the assumed maximum number of topics is much larger than the number of non-empty clusters. After initialization, the training algorithm of FGSDMM+ is the same as GSDMM, therefore the optimizations introduced by X-DMM is orthogonal to the differences between GSDMM and FGSDMM+. In the following sections, we describe the optimizations introduced by X-DMM by comparing with GSDMM.

X-DMM

Index-based optimization for sequential multiplication

DMM adopts the bag-of-words assumption in which it ignores the ordering information between words. Instead of assigning each word occurrence with an individual latent variable in topic models (e.g. LDA), all words in a document share the same latent variable (the mixture component assignment) in DMM model. However, as discussed in (Yin and Wang 2014; Yin and others 2016), the time complexity of GSDMM algorithm and its following improvement FGSDMM+ is still proportional to the length of documents. This leads to suboptimal training efficiency, especially for long text datasets.

We observe that, during the training of GSDMM model, the sequential multiplication of Equation 1 takes the vast majority of computation time. Since multiplication will be computed for each word occurrence, the time complexity of sampling for each document will be proportional to the length of document. For sequential multiplication with form like $\prod_{i=1}^{n}(x + N + i)$ (x is a decimal, N is an integer), if the value of x remains unchanged for all sequential multiplications during the training, then we can build the following index to accelerate the computation.

Figure 1: Index for sequential multiplication optimization, $a_i = \prod_{j=0}^{i}(x + j)$

The value of ith element in the index is $a_i = \prod_{j=0}^{i}(x+j)$. Once the index is built, the computation of $\prod_{i=1}^{n}(x+N+i)$ can be done by a single division $\prod_{i=1}^{n}(x+N+i) = \frac{a_{N+n}}{a_N}$, which reduces the time complexity from $O(n)$ to $O(1)$.

Table 2: Comparison of indexing time and GSDMM training time (milliseconds)

dataset	20ng	ohsumed	QA	Reuters
K	20	23	39	116
timeIndexing	196	375.3	391	64.3
timeTraining	43305	70760	29147	12210

If we add a restriction that only the symmetric Dirichlet prior can be used, that is, let $\overrightarrow{\alpha}$ and $\overrightarrow{\beta}$ to be vectors with same value in each dimension. Then the form of Equation 1 changes to Equation 2, which satisfies the requirement that the decimal remains unchanged for all sequential multiplications operations, thus we can build indices to achieve accelerations.

$$p(z_i = k|\overrightarrow{\mathbf{z}}_{\neg i}) \propto (m_{k,\neg i} + \alpha) \cdot \frac{\prod_{w \in \overrightarrow{d_i}} (\prod_{j=1}^{N_{iw}} (n_{kw,\neg i} + \beta + j - 1))}{\prod_{j=1}^{N_i} (n_{k,\neg i} + T\beta + j - 1)} \tag{2}$$

Specifically, we need to build two indices for Equation 2, one for denominator with $a_i = \prod_{j=0}^{i}(T\beta + j)$ and one for numerator with $a_i = \prod_{j=0}^{i}(\beta + j)$. With this optimization, the computation time of Equation 2 can be reduced to $O(1)$ for denominator and $O(U)$ for numerator.

Certainly, we need extra computation time to build the indices. However, compared to the total model training time, the indexing time is trivial. We need $O(max(n_k))$ time to build the indices, which roughly equals to $\frac{D*\bar{L}}{K}$ in magnitude. In contrast, the time complexity of original GSDMM algorithm is $D * \bar{L} * K * I$ (I is the number of training iterations). Thus, it only takes roughly $\frac{1}{K^2*I}$ of model training time to build the indices. Table 2 shows the comparison of indexing time and model training time (50 iterations) on several real-world datasets.

Metropolis-Hastings sampling

The computational time of sampling each document in GSDMM is proportional to the number of clusters. For large datasets, the number of clusters tends to be large, which requires more computational time to sample a single document. We observe that during the training of DMM model, the model will quickly converge to a state which is near convergence. However, it will take much more iterations for the model to finally converge. In this process, although the clustering accuracy of the model is stably increasing, the change of the model parameters (i.e. the word distribution in each cluster, φ_{kt}) in each iteration is very small. The training of GSDMM algorithm is agnostic of this manner, and the running time of each iteration remains the same.

We leverage the fact that in most of the training iterations the word distribution in each cluster varies very slowly, and thus we introduce a Metropolis-Hastings (MH) algorithm to accelerate the sampling process. MH algorithm (Chib and Greenberg 1995) is a fast sampling method introduced in statistics. Here we briefly describe the MH algorithm.

Algorithm 2 The Metropolis-Hastings algorithm

```
Draw initial sample x^0 ~ q(x)
for i = 1 to M do
    Draw x^cand ~ q(x^cand | x^{i-1})
    if RandUnif(1) < min(1, p(x^cand)*q(x^{i-1}|x^cand) / p(x^{i-1})*q(x^cand|x^{i-1})) then
        x^i = x^cand
    else
        x^i = x^{i-1}
    end if
end for
return x^i
```

Let $p(x)$ be the true target distribution we want to sample from. MH algorithm uses a proposal distribution $q(x)$, which is easier to be sampled than $p(x)$, to build a Markov Chain. By repeated generating samples from proposal distribution $q(x^i|x^{i-1})$ at step i, and updating the state with an acceptance rate $min(1, \frac{p(x^i)*q(x^{i-1}|x^i)}{p(x^{i-1})*q(x^i|x^{i-1})})$, the samples will converge to $p(x)$ in certain steps (Levin, Peres, and Wilmer 2009). Algorithm 2 demonstrates the details of MH algorithm. M is a hyperparameter named $MHSteps$. The closer the proposal distribution is to the target distribution, the smaller the $MHSteps$ needed for them to mix.

During the training of DMM model, the target distribution is the conditional distribution $p(z_i = k|\overrightarrow{\mathbf{z}}_{\neg i}, \overrightarrow{\mathbf{d}})$ given by Equation 2, which is expensive to draw ($O(K * U)$ time complexity). However, as $p(z_i = k|\overrightarrow{\mathbf{z}}_{\neg i}, \overrightarrow{\mathbf{d}})$ varies very slowly in each iteration, we can save $p(z_i = k|\overrightarrow{\mathbf{z}}_{\neg i}, \overrightarrow{\mathbf{d}})$ which occurs n iterations ago and use it as the proposal distribution. Note that directly using the latest $p(z_i = k)$ in last iteration as the proposal distribution is infeasible. This is because $O(K * U)$ complexity is needed for calculating the latest $p(z_i = k)$ for each k, which is actually the same as calculating $p(z_i = k)$ for current iteration. In the X-DMM's case, the proposal distribution is refreshed every n ($n \approx K$) iterations, which reduces the amortized cost for calculating proposal distribution to $O(U)$ complexity.

Drawing samples from the proposal distribution is much cheaper ($O(K)$ time complexity) as we don't need to compute the value of $p(z_i = k)$ for each k again. Moreover, since the proposal distribution and target distribution are close enough when the model is nearly-to-convergence, in practise we can effectively set $M = 1$ to make MH work. In the process of calculating the acceptance rate, the time complexity of calculating $\frac{p(x^{cand})*q(x^{i-1}|x^{cand})}{p(x^{i-1})*q(x^{cand}|x^{i-1})}$ is $O(U)$. Thus the complexity of MH algorithm for each iteration is $O(K + U)$.

In MH sampling algorithm, the proposal distribution can be used in n iterations. When n is quite big (e.g. $n = K$), we can use Walker sampling (Marsaglia et al. 2004) to furthur optimize the sampling process. Walker sampling is an efficient sampling method for drawing n samples from a multi-

Table 3: Speed-up of MH sampling on ohsumed dataset under different number of clusters

K	20	80	160	240
time (s) per iter (Gibbs)	2.04	7.60	16.32	23.5
#iterations (Gibbs)	30	38	45	48
time (s) per iter (MH)	0.456	0.493	0.534	0.54
#iterations (MH)	80	324	523	738

nomial distribution with K possible values in $O(n+K)$ time (averagely $O(1)$ for each draw when n is large). Therefore the time complexity of MH sampling for each iteration can be reduced from $O(K+U)$ to $O(U)$.

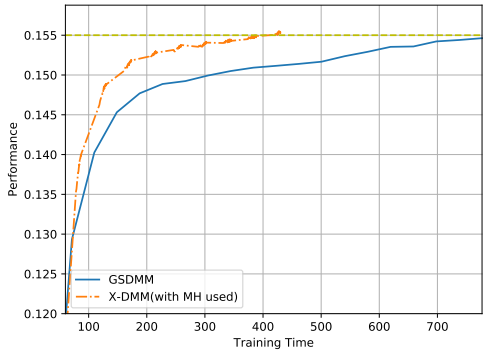

Figure 2: Comparison of convergence curves

Table 3 lists the average running time of an iteration and the number of running iterations for reaching convergence for both MH sampling and original Gibbs sampling algorithm. By using the MH algorithm, the time for each iteration decreases drastically, and the speed-up ratio is proportional to the number of clusters. However, since there are still differences between the proposal distribution and the target distribution (especially when n is assumed to be large), it will take much more iterations for the model to converge. Therefore the speed-up of convergence is much smaller than the speed-up of running time of a single iteration. Fig 2 shows the convergence curve of the experiments on ohsumed dataset (K=200), and we can see that the convergence rate has been accelerated substantially by MH algorithm.

Parallel training

For large datasets the training time will be very long, and we need to resort to parallel training for acceleration. However, the training of GSDMM uses a collapsed Gibbs sampler, which is a strictly sequential algorithm. The sampling result of one document will affect the sampling distribution of the next document, and thus can not be parallelized.

Algorithm 3 Parallel training of DMM

1: Initialize: for each document $\overrightarrow{d_i}$ in the corpus, randomly assign a cluster number z_i.
2: **for** $k = 1$ to K **do**
3: Draw $\varphi \sim p(\varphi|\overrightarrow{z}, \overrightarrow{d}, \overrightarrow{\alpha}, \overrightarrow{\beta})$
4: **end for**
5: **for** $i = 1$ to D **do**
6: Draw $z_i \sim p(z_i = k|\Phi, \overrightarrow{d}_i, \overrightarrow{\alpha}, \overrightarrow{\beta})$
7: **end for**
8: Repeat Step 2 – Step 7 until convergence.
9: Output the cluster assignments z_i of all documents.

We try to use an uncollapsed Gibbs sampler to eliminate the dependencies between the sampling results of documents in a training iteration, and therefore obtain an embarrassingly parallel training algorithm. The core idea is to sample not only the cluster assignment of each document, but also the word distribution of each component (i.e. each cluster). Algorithm 3 gives the details.

In the equation, $p(\varphi|\overrightarrow{z}, \overrightarrow{d}, \overrightarrow{\alpha}, \overrightarrow{\beta}) = Dirichlet(\varphi|\overrightarrow{n} + \overrightarrow{\beta})$, $p(z_i = k|\Phi, \overrightarrow{d}_i, \overrightarrow{\alpha}, \overrightarrow{\beta}) \propto p(z_i = k|\overrightarrow{z}, \overrightarrow{\alpha}) \cdot p(\overrightarrow{d}_i|z_i = k, \Phi_k)) \propto (m_k + \alpha_k) \cdot \prod_{w \in \overrightarrow{d_i}} (\varphi_{kw})^{N_{iw}}$. In practice we observe that using the posterior mean of φ instead of direct sampling it can accelerate the convergence of model training, and the posterior mean of φ is $\mathbb{E}(\varphi_{kt}) = \frac{n_{kt}+\beta_t}{n_k+\sum_{t=1}^{T}\beta_t}$.

The core computation of Algorithm 3 (Step 2 – Step 7) is embarrassingly parallel. As the datasets get larger, the amount of parallelism available in the algorithm increases. For large datasets, the efficiency speed up is nearly linear to the number of processors. We implement a CUDA program for parallel X-DMM with Nvidia GTX 1080. Fig 3 shows the scalability curve on 20ng and ohsumed datasets, and the speedup ratio is nearly proportional to the processor numbers.

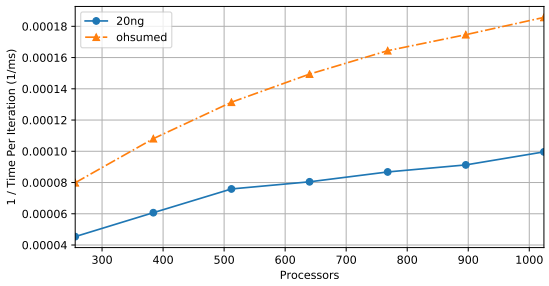

Figure 3: Scalability of parallel X-DMM

Experiments

Experimental Setup

The experiments are conducted on a PC with Intel CPU i5-7400 and Nvidia GPU GTX-1080. We use four real world

Table 4: Datasets description

dataset	K	D	avgDocLength	T
20ng	20	18846	138.852	181755
ohsumed	23	56984	114.884	92135
QA	39	2337	2968.61	264235
Reuters	116	11476	102.734	39600

datasets, 20ng[1], QA[2], and ohsumed[3], and Reuters[4].The statistics of each dataset are listed in Table 4.

Clustering Accuracy

To evaluate the clustering results, a widely used metric is the Normalized Mutual Information (NMI) (Strehl and Ghosh 2003). NMI is a normalization of the Mutual Information (MI) score to scale the results between zero and one (higher score indicates better clustering quality), the NMI is formally defined as follows:

$$NMI = \frac{\sum_{h,l} n_{h,l} log(\frac{n \cdot n_{h,l}}{n_h \cdot n_l})}{\sqrt{(\sum_h n_h log\frac{n_h}{n})(\sum_l n_l log\frac{n_l}{n})}}$$

Here n_h is the number of documents with cluster label h, n_l is the number of documents with cluster assignment l, and $n_{h,l}$ is the number of documents with cluster label h and cluster assignment l.

Since X-DMM adds a restriction that only symmetric Dirichlet prior can be used, we first test whether such restriction will influence the clustering accuracy. We run X-DMM ten times on four datasets with symmetric prior and asymmetric prior (index-based optimization is not employed when running asymmetric prior), and Fig 4 demonstrates the average NMI results of ten runs. To the best of our knowledge, there is no authoritative guide on how to choose a Dirichlet prior. In our experiments, we use the natural word frequency distribution as the asymmetric prior. The experiment results show that the symmetric prior restriction has little influence on the clustering accuracy.

We compare X-DMM with the following text clustering methods, K-Means (Aggarwal and Zhai 2012), LDA (Lu, Mei, and Zhai 2011), GSDMM (Yin and Wang 2014), and FGSDMM+ (Yin and others 2016),

1. **K-Means** K-means (Hartigan and Wong 1979) is a very popular and probably the most widely used method for clustering. Following (Aggarwal and Zhai 2012), we use cosine similarity as the similarity metric.
2. **LDA** Following (Lu, Mei, and Zhai 2011), We treat each of the topics found by LDA (Blei, Ng, and Jordan 2003) as a cluster and assign each document to the cluster with the highest value in its topic proportion vector. Similar to (Griffiths and Steyvers 2004), we set $\alpha = 50/K$ and $\beta = 0.1$.

[1]http://qwone.com/~jason/20Newsgroups/
[2]https://www.cs.cmu.edu/~ark/QA-data/
[3]http://davis.wpi.edu/xmdv/datasets/ohsumed
[4]https://www.kaggle.com/nltkdata/reuters

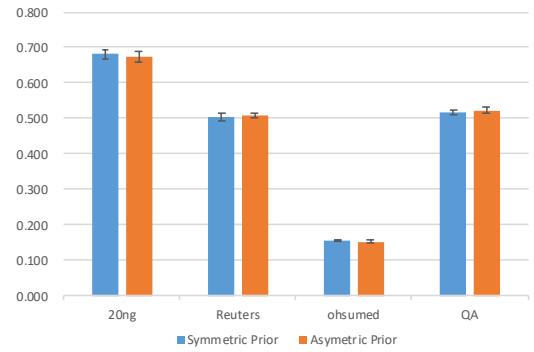

Figure 4: NMI score under symmetric Dirichlet prior and asymmetric Dirichlet prior

3. **GSDMM** This is the state-of-the-art text clustering method which adopts a collapsed Gibbs sampling training algorithm for the Dirichlet Multinomial Mixture model. Similar to (Yin and Wang 2014), we set $\alpha = 0.1$ and $\beta = 0.1$ for GSDMM.
4. **FGSDMM+** FGSDMM+ improves the training algorithm of GSDMM by introducing a online initialization scheme. Following (Yin and others 2016), we also set $\alpha = 0.1$ and $\beta = 0.1$ for FGSDMM+.

We compare each method's clustering accuracy on four datasets, and the results are shown in Table 5. We run each method 20 times on each dataset, and report the mean and the standard deviation of the NMI scores. We can see that the X-DMM obtains the similar NMI results with the existing state-of-the-art methods (GSDMM, FGSDMM+), and is much better than the baseline methods.

Efficiency

As X-DMM uses a different sampler, the running time of each iteration for X-DMM is much smaller than GSDMM. However, it takes more iterations for X-DMM to converge. Therefore, the running time for a fixed number of iterations would no longer fairly reflect the efficiency improvement of X-DMM. In this section, we compare the running time for each method to reach final convergence, which is a more fair-minded metric for training efficiency. The cluster numbers are set as K listed in Table 4. Fig 5 shows the experimental results, and we can see that X-DMM is much faster than GSDMM and FGSDMM+, and the parallel implementation can bring a further speedup.

We observe that, for larger datasets the X-DMM can achieve more acceleration. From this, we speculate that if larger labeled datasets are available (unfortunately, to the best of our knowledge there are no very large public labeled datasets for text clustering), the X-DMM can obtain more efficiency speedup. Though in real world applications, text clustering will be used to process very large data without labels, for calculating the NMI score we still need labels.

To test the efficiency promotion of X-DMM on large unlabeled datasets, we need another metric to evaluate the

Table 5: NMI results comparison of each methods (average results of 20 trials)

Datasets	K	**K-Means**	**LDA**	**GSDMM**	**FGSDMM+**	**X-DMM**	**X-DMM parallel**
20ng	40	0.407±0.011	0.513±0.009	0.689±0.010	0.691±0.016	0.691±0.012	0.682±0.022
20ng	80	0.392±0.009	0.448±0.007	0.687±0.013	0.689±0.011	0.681±0.015	0.689±0.015
ohsumed	46	0.138±0.006	0.141±0.004	0.156±0.004	0.156±0.002	0.158±0.003	0.152±0.003
ohsumed	92	0.157±0.003	0.150±0.003	0.156±0.002	0.155±0.003	0.154±0.003	0.153±0.002
QA	78	0.427±0.010	0.453±0.006	0.514±0.012	0.512±0.012	0.516±0.009	0.513±0.006
QA	156	0.289±0.114	0.447±0.007	0.520±0.010	0.521±0.008	0.525±0.005	0.510±0.007
Reuters	232	0.471±0.003	0.478±0.003	0.506±0.011	0.508±0.008	0.510±0.010	0.498±0.010
Reuters	464	0.485±0.004	0.482±0.004	0.519±0.007	0.517±0.013	0.514±0.011	0.509±0.008

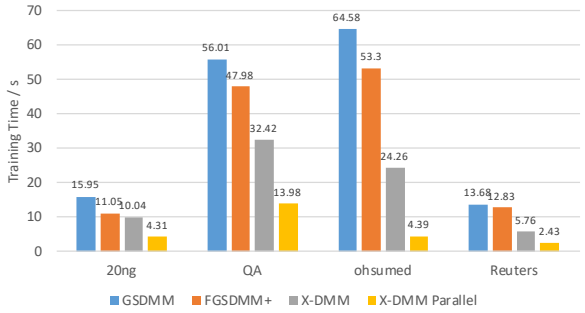

Figure 5: Running time for each method to convergence

convergence of model training without labels. Since the probabilistic generative process adopted by GSDMM, FGSDMM+ and XDMM are actually the same, here we can directly use *perplexity* as the metric of modeling quality.

Perplexity is used by convention in language modeling. It is actually the generative probability of test documents. Formally, for a set C of D documents, the perplexity is

$$perplexity(C) = \exp\{-\frac{\sum_{d=1}^{D} \log p(\overrightarrow{w}_d)}{\sum_{d=1}^{D} N_d}\} \qquad (3)$$

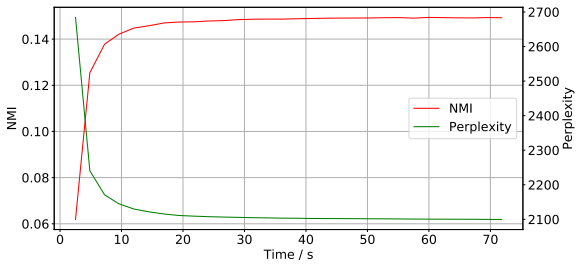

Figure 6: NMI and perplexity score during DMM model training

Fig 6 illustrates the NMI score and perplexity during GSDMM model training on ohsumed dataset. In this figure the curves of two metrics converge at a nearly same rate (lower perplexity indicates better modeling quality), which shows that the perplexity metric is capable to monitor the convergence of DMM model training on large unlabeled datasets.

Table 6: Comparison of running time (seconds) to reach the same perplexity on nytimes dataset (K=200)

Perplexity	3981	3941	3901	3881
GSDMM	440.6	726.2	1641.9	4162.5
FGSDMM+	401.1	703.3	1811.6	3423.3
X-DMM	268.6	369.2	565.8	849.9
X-DMM Parallel	78.9	121.4	170.4	262.3

For efficiency comparison on large unlabeled corpus, we run each method ten times on NYTimes article dataset[5] (300K documents, 102K distinct words, 331 average document length). Table 6 compares the efficiency of each method by presenting their average running time to a given value of perplexity. We can see that X-DMM converges much faster than GSDMM and FGSDMM+, especially in the nearly-to-convergence stages (4.0-4.8x speedup for reaching to 3881 perplexity), and the GPU parallel implementation of X-DMM can further accelerate the training by 10x times than GSDMM and FGSDMM+.

Conclusion

In this work, we present X-DMM, a highly efficient system for text clustering. X-DMM reduces the sampling complexity of GSDMM from $O(K * L)$ to $O(K * U)$ by leveraging a symmetric Dirichlet setting, and further reduces the sampling complexity from $O(K * U)$ to $O(U)$ in the nearly-to-convergence training stages by using a Metropolis-Hastings algorithm. X-DMM also introduces an uncollapsed Gibbs sampler to parallelize the model training. With these optimizations, X-DMM is highly efficient for long and large scale text clustering and can scale up almost linearly with increased number of processors.

Empirically, we show that X-DMM obtains substantial speed-up compared to existing state-of-the-art methods without clustering accuracy degradation. X-DMM also shows its good scalability. For large datasets, the parallel GPU implementation of X-DMM can achieve several folds in speed up than non-parallel implementation with only slight clustering accuracy degradation.

[5]https://archive.ics.uci.edu/ml/machine-learning-databases/bag-of-words/docword.nytimes.txt.gz

Acknowledgment

We would like to thank the anonymous reviewers for their helpful feedbacks. This work was supported by the National Natural Science Foundation of China (Grant No.61732004 and 61872207) and the National Key Research and Development Program of China (Grant No.2018YFB1004404 and 2018YFC0830900).

References

Aggarwal, C. C., and Zhai, C. 2012. A survey of text classification algorithms. In *Mining text data*. Springer. 163–222.

Anastasiu, D. C.; Tagarelli, A.; and Karypis, G. 2013. Document clustering: The next frontier.

Blei, D. M.; Griffiths, T. L.; and Jordan, M. I. 2010. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *Journal of the ACM (JACM)* 57(2):7.

Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3(Jan):993–1022.

Chib, S., and Greenberg, E. 1995. Understanding the metropolis-hastings algorithm. *The american statistician* 49(4):327–335.

Christopher, M. B. 2016. *PATTERN RECOGNITION AND MACHINE LEARNING*. Springer-Verlag New York.

Dhillon, I. S., and Modha, D. S. 2001. Concept decompositions for large sparse text data using clustering. *Machine learning* 42(1-2):143–175.

Elkan, C. 2006. Clustering documents with an exponential-family approximation of the dirichlet compound multinomial distribution. In *Proceedings of the 23rd international conference on Machine learning*, 289–296. ACM.

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

Hartigan, J. A., and Wong, M. A. 1979. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28(1):100–108.

Hofmann, T. 1999. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, 289–296. Morgan Kaufmann Publishers Inc.

Huang, R.; Yu, G.; Wang, Z.; Zhang, J.; and Shi, L. 2013. Dirichlet process mixture model for document clustering with feature partition. *IEEE Transactions on knowledge and data engineering* 25(8):1748–1759.

Jain, A. K. 2010. Data clustering: 50 years beyond k-means. *Pattern recognition letters* 31(8):651–666.

Levin, D. A.; Peres, Y.; and Wilmer, E. L. 2009. *Markov chains and mixing times*. American Mathematical Soc.

Li, A. Q.; Ahmed, A.; Ravi, S.; and Smola, A. J. 2014. Reducing the sampling complexity of topic models. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 891–900. ACM.

Lu, Y.; Mei, Q.; and Zhai, C. 2011. Investigating task performance of probabilistic topic models: an empirical study of plsa and lda. *Information Retrieval* 14(2):178–203.

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

Newman, D.; Asuncion, A.; Smyth, P.; and Welling, M. 2009. Distributed algorithms for topic models. *Journal of Machine Learning Research* 10(Aug):1801–1828.

Nigam, K.; McCallum, A. K.; Thrun, S.; and Mitchell, T. 2000. Text classification from labeled and unlabeled documents using em. *Machine learning* 39(2):103–134.

Paisley, J.; Wang, C.; Blei, D. M.; and Jordan, M. I. 2015. Nested hierarchical dirichlet processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37(2):256–270.

Papadimitriou, C. H.; Raghavan, P.; Tamaki, H.; and Vempala, S. 2000. Latent semantic indexing: A probabilistic analysis. *Journal of Computer and System Sciences* 61(2):217–235.

Park, H.-S., and Jun, C.-H. 2009. A simple and fast algorithm for k-medoids clustering. *Expert systems with applications* 36(2):3336–3341.

Strehl, A., and Ghosh, J. 2003. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research* 3(Dec):583–617.

Yin, J., et al. 2016. A text clustering algorithm using an online clustering scheme for initialization. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 1995–2004. ACM.

Yin, J., and Wang, J. 2014. A dirichlet multinomial mixture model-based approach for short text clustering. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 233–242. ACM.

Yu, G.; Huang, R.; and Wang, Z. 2010. Document clustering via dirichlet process mixture model with feature selection. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 763–772. ACM.

Zhao, Y., and Karypis, G. 2004. Empirical and theoretical comparisons of selected criterion functions for document clustering. *Machine Learning* 55(3):311–331.

Zhao, Y.; Karypis, G.; and Fayyad, U. 2005. Hierarchical clustering algorithms for document datasets. *Data mining and knowledge discovery* 10(2):141–168.