A Text Clustering Algorithm Using an Online Clustering Scheme for Initialization

Jianhua Yin[†], Jianyong Wang[†‡]
[†]Tsinghua National Laboratory for Information Science and Technology (TNList)
Department of Computer Science and Technology, Tsinghua University, Beijing, China
[‡]Jiangsu Collaborative Innovation Center for Language Ability, Jiangsu Normal University, Xuzhou 221009, China
jhyin12@gmail.com, jianyong@tsinghua.edu.cn

ABSTRACT

In this paper, we propose a text clustering algorithm using an online clustering scheme for initialization called FGSDMM+. FGSDMM+ assumes that there are at most K_{max} clusters in the corpus, and regards these K_{max} potential clusters as one large potential cluster at the beginning. During initialization, FGSDMM+ processes the documents one by one in an online clustering scheme. The first document will choose the potential cluster, and FGSDMM+ will create a new cluster to store this document. Later documents will choose one of the non-empty clusters or the potential cluster with probabilities derived from the Dirichlet multinomial mixture model. Each time a document chooses the potential cluster, FGSDMM+ will create a new cluster to store that document and decrease the probability of later documents choosing the potential cluster. After initialization, FGSDMM+ will run a collapsed Gibbs sampling algorithm several times to obtain the final clustering result. Our extensive experimental study shows that FGSDMM+ can achieve better performance than three other clustering methods on both short and long text datasets.

Keywords

Text Clustering; Gibbs Sampling; Dirichlet Multinomial Mixture

1. INTRODUCTION

Text clustering [1] is a widely studied problem with many applications such as document organization, summarization, and browsing. In [27], we introduced a collapsed Gibbs Sampling algorithm for the Dirichlet Multinomial Mixture model (GSDMM) for text clustering. GSDMM represents each cluster as a large document combined by the documents in the cluster and records the size of each cluster. The probability of a document belonging to each cluster is proportional to the size of the cluster and the frequency of each word of the document in the cluster. In this way, GSDMM considers both the size of the cluster and the similarity between the document and the cluster when assigning a document to a cluster. GSDMM assumes that there are at most K_{max} clusters in the corpus. This assumption is weaker than assuming the true number of clusters in the corpus. Besides, this assumption can certainly be right when we set the assumed maximum number of clusters K_{max} to be the number of documents in the corpus. During initialization, GSDMM randomly assigns the documents to K_{max} clusters. Then GSDMM traverses the documents for several iterations. For each document, GSDMM re-assigns it to one of the K_{max} clusters according to the probability of the document belonging to each cluster derived from the Dirichlet multinomial mixture (DMM) model [16]. In [27], we found that GSDMM can achieve good performance as long as the assumed maximum number of clusters K_{max} is larger than the true number of clusters. However, it is difficult to set a proper maximum number of clusters for GSDMM as we do not know the true number. As a result, we may have to choose a really large K_{max} to ensure safety which will result in the complexity of GSDMM to be large. In addition, GSDMM randomly assigns the documents to K_{max} clusters during initialization, which will introduce noise into clustering. Because the documents in an initial cluster have high probability of coming from different classes.

An observation is that when the assumed maximum number of clusters K_{max} is much larger than the true number of clusters (such as K_{max} is set to be the number of documents in the corpus), the number of non-empty clusters K_{non} drops quickly when we run GSDMM. Actually, the probability of a document choosing each empty cluster is the same, and there is no difference which empty cluster the document chooses. Therefore, we can regard all empty clusters as a potential cluster, and the probability of a document choosing this potential cluster is $(K_{max} - K_{non})$ times the probability of that document choosing a specific empty cluster. In each iteration, each document can choose one of the K_{non} non-empty clusters or the potential cluster according to its probability of belonging to the $K_{non} + 1$ clusters. We call this new text clustering algorithm as Fast GSDMM (abbr. to FGSDMM). The time complexity of FGSDMM is linear to the number of non-empty clusters.

Although FGSDMM can reduce the time complexity of GSDMM, its space complexity is still linear to the assumed maximum number of clusters. Besides, the initialization step of FGSDMM is the same as GSDMM, which will in-

KDD '16, August 13–17, 2016, San Francisco, CA, USA.

DOI: http://dx.doi.org/10.1145/2939672.2939841

troduce noise into clustering. An observation is that the initialization approach of FGSDMM wastes much information, because it does not consider the differences between the clusters. Actually, we can estimate the probability of a document choosing each cluster as we already know the current representation of the clusters. Under the above inspiration, we propose a new initialization approach for FGSDMM using an online clustering scheme called FGSDMM+. The space complexity of FGSDMM+ is linear to the number of non-empty clusters, because it only needs to store the information of the non-empty clusters. Besides, FGSDMM+ introduces less noise into clustering, because it assigns each document to a cluster according to the probability of the document belonging to each cluster.

In the experimental study, we compared FGSDMM+ with GSDMM [27], K-means [7], and LDA [5]. We found that FGSDMM+ can achieve better performance than these methods on both short and long text datasets. We compared the speed of FGSDMM+, FGSDMM, and GSDMM with different assumed maximum number of clusters K_{max}. We found that the speed of GSDMM is approximately linear to the assumed maximum number of clusters K_{max}, and the speed of FGSDMM+ gets almost stable when the assumed maximum number of clusters is larger than the true number of clusters. We also investigated the influence of the parameters to the number of clusters found by FGSDMM+ and the performance of FGSDMM+.

The contributions of this paper are summarized as follows.

- We proposed a new Gibbs sampling algorithm for the Dirichlet multinomial mixture model for text clustering whose time complexity is linear to the number of non-empty clusters.
- We proposed a text clustering algorithm using an online clustering scheme for initialization which has low time and space complexity and can detect the number of clusters automatically.
- We conducted extensive experiments on both short and long text datasets. Our algorithm can achieve better performance than three other clustering methods on all datasets.

The remainder of this paper is organized as follows. In Section 2, we first introduce the Dirichlet Multinomial Mixture (DMM) model, then we introduce the GSDMM algorithm. In Section 3, we propose the FGSDMM algorithm whose time complexity is linear to the number of non-empty clusters. In Section 4, we propose the FGSDMM+ algorithm which uses an online clustering scheme for initialization and can detect the number of clusters automatically. Section 5 describes the design of experiments to evaluate the performance of our algorithm and the influence of parameters to the algorithm. In Section 6, we review the related work of text clustering. We finally present conclusions and future work in Section 7.

2. BACKGROUND

In this section, we will introduce the Dirichlet Multinomial Mixture (DMM) model [16] and the GSDMM algorithm we proposed in [27]. In the next two sections, we will point out the drawbacks of GSDMM and propose two new clustering algorithms to cope with these drawbacks.

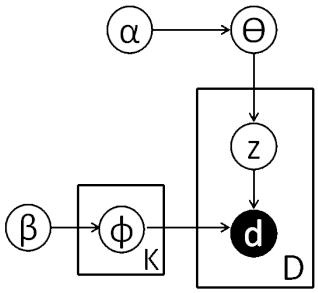


Figure 1: Graphical model of DMM.

K_{max}	assumed maximum number of clusters
K_{non}	number of non-empty clusters
V	size of the vocabulary
D	number of documents in the corpus
I	number of iterations
$\vec{d}$	documents in the corpus
$\vec{z}$	cluster assignments of each document
m_z	number of documents in cluster z
n_z	number of words in cluster z
n_z^w	frequency of word w in cluster z
N_d	number of words in document d
N_d^w	frequency of word w in document d
α	pseudo number of documents in each cluster
β	pseudo frequency of each word in each cluster

Table 1: Notations

2.1 The DMM Model

The Dirichlet Multinomial Mixture (DMM) model [16] is a probabilistic generative model for documents, and embodies two assumptions about the generative process: (1) the documents are generated by a mixture model [15], and (2) there is a one-to-one correspondence between mixture components and clusters. The graphical representation of the Dirichlet Multinomial Mixture (DMM) model is shown in Figure 1 (Table 1 shows the notations used in this paper), which is equivalent to the following generative process:

$$\Theta|\alpha \sim Dir(\alpha) \tag{1}$$

$$z_d|\Theta \sim Mult(\Theta) \qquad d = 1, ..., D \tag{2}$$

$$\Phi_k|\beta \sim Dir(\beta) \qquad k = 1, ..., K \tag{3}$$

$$d|z_d, \{\Phi_k\}_{k=1}^K \sim p(d|\Phi_{z_d}) \tag{4}$$

Here, "$X \sim S$" means "X is distributed according to S", so the right side is a specification of distribution. When generating document d, the Dirichlet Multinomial Mixture (DMM) model first selects a mixture component (cluster) z_d for document d according to the mixture weights (weights of clusters), Θ, in Equation 2. Then document d is generated by the selected mixture component (cluster) from distribution $p(d|\Phi_{z_d})$ in Equation 4. The weight vector of the clusters, Θ, is generated by a Dirichlet distribution with a hyper-parameter α, as in Equation 1. The cluster parameters Φ_z are also generated by a Dirichlet distribution with a hyper-parameter β, as in Equation 3.

In this paper, the probability of document d generated by

cluster z_d is defined as follows:

$$p(d|\Phi_{z_d}) = \prod_{w \in d} Mult(w|\Phi_{z_d}) \quad (5)$$

Here, we make the Naive Bayes assumption: the words in a document are generated independently when the document's cluster assignment z_d is known. We also assume that the probability of a word is independent of its position within the document.

2.2 The GSDMM Algorithm

In this part, we give the derivation of the collapsed Gibbs sampling algorithm for the Dirichlet Multinomial Mixture (DMM) model, which is different from the one presented in [27]. The documents $\vec{d} = \{d_i\}_{i=1}^{D}$ are observed and the cluster assignments $\vec{z} = \{z_i\}_{i=1}^{D}$ are latent. The conditional probability of document d choosing cluster z given the information of other documents and their cluster assignments can be factorized as follows:

$$p(z_d = z|\vec{z}_{\neg d}, \vec{d}, \alpha, \beta)$$
$$\propto p(z_d = z|\vec{z}_{\neg d}, \vec{d}_{\neg d}, \alpha, \beta) p(d|z_d = z, \vec{z}_{\neg d}, \vec{d}_{\neg d}, \alpha, \beta) \quad (6)$$
$$\propto p(z_d = z|\vec{z}_{\neg d}, \alpha) p(d|z_d = z, \vec{d}_{z,\neg d}, \beta) \quad (7)$$

Here, we use the Bayes Rule in Equation 6, and apply the properties of D-Separation [4] in Equation 7.

The first term in Equation 7 indicates the probability of document d choosing cluster z when we know the cluster assignments of other documents. The second term in Equation 7 can be considered as a predictive probability of document d given $\vec{d}_{z,\neg d}$, i.e., the other documents currently assigned to cluster z.

The first term in Equation 7 can be derived as follows:

$$p(z_d = z|\vec{z}_{\neg d}, \alpha)$$
$$= \int p(z_d = z, \Theta|\vec{z}_{\neg d}, \alpha) d\Theta \quad (8)$$
$$= \int p(\Theta|\vec{z}_{\neg d}, \alpha) p(z_d = z|\vec{z}_{\neg d}, \Theta, \alpha) d\Theta \quad (9)$$
$$= \int p(\Theta|\vec{z}_{\neg d}, \alpha) p(z_d = z|\Theta) d\Theta \quad (10)$$
$$= \int Dir(\Theta|\vec{m}_{\neg d} + \vec{\alpha}) Mult(z_d = z|\Theta) d\Theta \quad (11)$$
$$= \int \frac{1}{\Delta(\vec{m}_{\neg d} + \vec{\alpha})} \Theta_z \prod_{k=1, k\neq z}^{K} \Theta_k^{m_{k,\neg d} + \alpha - 1} d\Theta \quad (12)$$
$$= \frac{\Delta(\vec{m} + \vec{\alpha})}{\Delta(\vec{m}_{\neg d} + \vec{\alpha})} \quad (13)$$
$$= \frac{\prod_{k=1}^{K} \Gamma(m_k + \alpha)}{\Gamma(\sum_{k=1}^{K}(m_k + \alpha))} \frac{\Gamma(\sum_{k=1}^{K}(m_{k,\neg d} + \alpha))}{\prod_{k=1}^{K} \Gamma(m_{k,\neg d} + \alpha)} \quad (14)$$
$$= \frac{\Gamma(m_{z,\neg d} + \alpha + 1)}{\Gamma(m_{z,\neg d} + \alpha)} \frac{\Gamma(D - 1 + K\alpha)}{\Gamma(D + K\alpha)} \quad (15)$$
$$= \frac{m_{z,\neg d} + \alpha}{D - 1 + \alpha} \quad (16)$$

Here, Equation 8 exploits the Sum Rule of Probability [4]. We use the Product Rule of Probability [4] in Equation 9 and apply the properties of D-Separation in Equation 10. The posterior distribution of Θ is a Dirichlet distribution, because we assumed Dirichlet prior $Dir(\Theta|\alpha)$ for the Multinomial distribution $Mult(\vec{z}|\Theta)$. In Equation 13, we adopt the Δ function used in [11], which is defined as $\Delta(\vec{\alpha}) = \frac{\prod_{k=1}^{K} \Gamma(\alpha)}{\Gamma(\sum_{k=1}^{K} \alpha)}$. Using the property of Γ function: $\Gamma(x+1) = x\Gamma(x)$, we can get Equation 16 from Equation 15. In Equation 16, $m_{z,\neg d}$ is the number of documents in cluster z without considering document d, and D is the total number of documents in the dataset. Equation 16 indicates that document d will tend to choose larger clusters when we only consider the cluster assignments of the other documents.

Then, we derive the second term in Equation 7 as follows:

$$p(d|z_d = z, \vec{d}_{z,\neg d}, \beta)$$
$$= \int p(d, \Phi_z|z_d = z, \vec{d}_{z,\neg d}, \beta) d\Phi_z \quad (17)$$
$$= \int p(\Phi_z|z_d = z, \vec{d}_{z,\neg d}, \beta) p(d|\Phi_z, z_d = z, \vec{d}_{z,\neg d}, \beta) d\Phi_z \quad (18)$$
$$= \int p(\Phi_z|\vec{d}_{z,\neg d}, \beta) p(d|\Phi_z, z_d = z) d\Phi_z \quad (19)$$
$$= \int Dir(\Phi_z|\vec{n}_{z,\neg d} + \beta) \prod_{w \in d} Mult(w|\Phi_z) d\Phi_z \quad (20)$$
$$= \int \frac{1}{\Delta(\vec{n}_{z,\neg d} + \beta)} \prod_{t=1}^{V} \Phi_{z,t}^{n^t_{z,\neg d} + \beta - 1} \prod_{w \in d} \Phi_{z,w}^{n^w_d} d\Phi_z \quad (21)$$
$$= \frac{\Delta(\vec{n}_z + \beta)}{\Delta(\vec{n}_{z,\neg d} + \beta)} \quad (22)$$
$$= \frac{\prod_{t=1}^{V} \Gamma(n^t_z + \beta)}{\Gamma(\sum_{t=1}^{V}(n^t_z + \beta))} \frac{\Gamma(\sum_{t=1}^{V}(n^t_{z,\neg d} + \beta))}{\prod_{t=1}^{V} \Gamma(n^t_{z,\neg d} + \beta)} \quad (23)$$
$$= \frac{\prod_{w \in d} \prod_{j=1}^{N^w_d}(n^w_{z,\neg d} + \beta + j - 1)}{\prod_{i=1}^{N_d}(n_{z,\neg d} + V\beta + i - 1)} \quad (24)$$

Here, Equation 17 exploits the Sum Rule of Probability [4]. We use the Product Rule of Probability in Equation 18 and apply the properties of D-Separation [4] to obtain Equation 19. The posterior distribution of Φ_z is a Dirichlet distribution, because we assumed Dirichlet prior $Dir(\Phi_z|\beta)$ for the Multinomial distribution $Mult(\vec{d}_z|\Phi_z)$. Because the Γ function has the following property: $\frac{\Gamma(x+m)}{\Gamma(x)} = \prod_{i=1}^{m}(x+i-1)$, we can get Equation 24 from Equation 23. In Equation 24, N^w_d and N_d are the number of occurrences of word w in document d and the total number of words in document d, respectively, and $N_d = \sum_{w \in d} N^w_d$. Besides, $n^w_{z,\neg d}$ and $n_{z,\neg d}$ are the number of occurrences of word w in cluster z and the total number of words in cluster z without considering document d, respectively, and $n_{z,\neg d} = \sum_{w=1}^{V} n^w_{z,\neg d}$. We can notice that Equation 24 actually evaluates some kind of similarity between document d and cluster z, and document d will tend to choose a cluster whose documents share more words with it.

Finally, we have the probability of document d choosing cluster z_d given the information of other documents and their cluster assignments as follows:

$$p(z_d = z|\vec{z}_{\neg d}, \vec{d}, \alpha, \beta)$$
$$\propto (m_{z,\neg d} + \alpha) \frac{\prod_{w \in d} \prod_{j=1}^{N^w_d}(n^w_{z,\neg d} + \beta + j - 1)}{\prod_{i=1}^{N_d}(n_{z,\neg d} + V\beta + i - 1)} \quad (25)$$

The first part of Equation 25 relates to Rule 1 of GSDMM (i.e., choose a cluster with more documents), as it will have larger value when $m_{z,\neg d}$ (number of documents in cluster

z without considering document d) is larger. This is also known as the "richer gets richer" property, which will lead larger clusters to get larger [24]. The second part of Equation 25 relates to Rule 2 of GSDMM (i.e., choose a cluster which is more similar with the current document), which actually defines the similarity between the document and the cluster. It is a product of N_d parts that correspond to the N_d words in document d. For each word w in document d, the corresponding part measures the the frequency of word w in cluster z. When a cluster has more documents that share same words with document d, the second part of Equation 25 will be larger, and document d will be more likely to choose that cluster.

Most text clustering algorithms [1] represent documents with the vector space model [21], in which each document is represented with a vector in length of the size of the vocabulary. Differently, GSDMM represents each document with its words and the frequency of each word in the document. GSDMM represents each cluster as a large document combined by the documents in the cluster and records the size of each cluster. In detail, GSDMM uses three count variables to record the information of each cluster: n_z^w (frequency of word w in cluster z), n_z (number of words in cluster z), m_z (number of documents in cluster z).

GSDMM assumes that there are at most K_{max} clusters in the corpus. This assumption is weaker than assuming the true number of clusters in the corpus. Besides, this assumption can certainly be right when we set the assumed maximum number of clusters K_{max} to be the number of documents in the corpus. During initialization, GSDMM randomly assigns the documents to K_{max} clusters. Then GSDMM traverses the documents for several iterations. In each iteration, GSDMM re-assigns each document to one of the K_{max} clusters according to the probability of the document belonging to each cluster computed from Equation 25. Because the documents tend to choose clusters with more documents, large clusters will tend to get larger and small clusters will tend to get smaller. After several iterations, some clusters will become empty, as a result, GSDMM can detect the number of clusters automatically.

3. THE FGSDMM ALGORITHM

In [27], we found that GSDMM can achieve good performance as long as the assumed maximum number of clusters K_{max} is larger than the true number of clusters. However, it is difficult to set a proper maximum number of clusters for GSDMM as we do not know the true number of clusters in the dataset beforehand. As a result, we may have to choose a really large assumed maximum number of clusters K_{max} to ensure safety which will result in the complexity of GSDMM to be large.

An observation is that sampling a cluster from the K_{max} clusters for each document wastes much time when the assumed maximum number of clusters K_{max} is much larger than the true number of clusters, because most of the clusters will become empty quickly. For example, when we set K_{max} to be the number of documents in the corpus, the number of non-empty clusters K_{non} will decrease really quickly when we run GSDMM.

Actually, the probability of a document choosing each empty cluster is the same, and there is no difference which empty cluster the document chooses. Therefore, we can regard all empty clusters as a potential cluster, and the probability of a document choosing this potential cluster is $(K_{max} - K_{non})$ times the probability of that document choosing a specific empty cluster. In each iteration, each document will choose one of the K_{non} non-empty clusters or the potential cluster according to its probability of belonging to the $K_{non} + 1$ clusters.

Algorithm 1: FGSDMM

Data: Documents $\vec{d}$, Maximum number of clusters K_{max}.
Result: Cluster assignments of the documents $\vec{z}$.
begin
  //Initialization
  $K_{non} \leftarrow K_{max}$
  Zero all count variables for each cluster.
  for *each document* $d \in [1, D]$ **do**
    Sample cluster index z for document d as one of the K_{max} clusters with equal probability.
    Assign document d to cluster z and update its count variables.
  //Gibbs Sampling
  for $i \in [1, I]$ **do**
    for *each document* $d \in [1, D]$ **do**
      Record the current cluster of d: $z \leftarrow z_d$
      Remove document d from cluster z and update its count variables.
      if $m_z == 0$ **then**
        $K_{non} \leftarrow K_{non} - 1$
        Re-arrange cluster indices so that $1, ..., K_{non}$ are non-empty clusters;
      Compute the probability of document d choosing each of the K_{non} non-empty clusters or the potential cluster with Equation 25 and Equation 26, respectively.
      Sample cluster index z for document d according to the above $K_{non} + 1$ probabilities.
      if $z == K_{non} + 1$ **then**
        $K_{non} \leftarrow K_{non} + 1$
      Assign document d to cluster z and update its count variables.

The probability of a document choosing one of the K_{non} non-empty clusters is shown in Equation 25. We denote the potential cluster that represent the $K_{max} - K_{non}$ empty clusters as $K_{non} + 1$, and the probability of a document choosing the potential cluster is as follows:

$$p(z_d = K_{non} + 1 | \vec{z}_{\neg d}, \vec{d}) \propto \alpha(K_{max} - K_{non}) \frac{\prod_{w \in d} \prod_{j=1}^{N_d^w} (\beta + j - 1)}{\prod_{i=1}^{N_d} (V\beta + i - 1)} \quad (26)$$

We call this new text clustering algorithm as FGSDMM (abbr. for Fast GSDMM) which is shown in Algorithm 1. The main difference between FGSDMM and GSDMM is in the way of assigning a document to a cluster. GSDMM samples a cluster from the K_{max} clusters for each document in each iteration. This means the time complexity of GSDMM is linear to the assumed maximum number of clusters K_{max}.

Different from GSDMM, the time complexity of FGSDMM is linear to the number of non-empty clusters.

4. THE FGSDMM+ ALGORITHM

Although FGSDMM can reduce the time complexity of GSDMM, its space complexity is still linear to the assumed maximum number of clusters K_{max}. Because FGSDMM needs to store the frequency of each word in each cluster and the size of each cluster. For text datasets, the size of the vocabulary is in the order of 10^5, which means FGSDMM needs much memory when the assumed maximum number of clusters K_{max} is really large.

In addition, FGSDMM randomly assigns the documents to K_{max} clusters during initialization, which will introduce noise into clustering. This is because the documents in an initial cluster have high probability of coming from different classes.

During initialization, FGSDMM traverses the documents and randomly assigns each document to one of the K_{max} clusters with equal probability. In detail, the first document will choose one of the K_{max} empty clusters with equal probability. Later documents will choose one of the K_{non} non-empty clusters or one of the $K_{max} - K_{non}$ empty clusters with equal probability.

Algorithm 2: FGSDMM+

Data: Documents $\vec{d}$, Maximum number of clusters K_{max}.
Result: Cluster assignments of the documents $\vec{z}$.
begin
  //Initialization
  $K_{non} \leftarrow 0$
  for *each document* $d \in [1, D]$ **do**
    Compute the probability of document d choosing each of the K_{non} non-empty clusters or the potential cluster with Equation 25 and Equation 26, respectively.
    Sample cluster index z for document d according to the above $K_{non} + 1$ probabilities.
    if $z <= K_{non}$ **then**
      Assign document d to cluster z and update its count variables.
    else
      $K_{non} \leftarrow K_{non} + 1$
      Create a cluster to store document d and initialize its count variables.
  //Gibbs Sampling
  The Gibbs sampling step of FGSDMM+ is similar to FGSDMM, except that FGSDMM+ needs to create a new cluster to store a document when it chooses the potential cluster.

An observation is that the initialization approach of FGSDMM wastes much information, because it does not consider the differences between the clusters. Actually, we can estimate the probability of a document choosing each cluster as we already know the current representation of the clusters. Under the above inspiration, we propose a new initialization approach for FGSDMM using an online clustering scheme. We call this new text clustering algorithm FGSDMM+ which is shown in Algorithm 2.

Just like FGSDMM, FGSDMM+ assumes that there are at most K_{max} clusters in the corpus. Different from FGSDMM, FGSDMM+ does not allocate space for these K_{max} clusters at the beginning. Instead, it regards the K_{max} potential clusters as one large potential cluster.

During initialization, FGSDMM+ processes the documents one by one in an online clustering scheme. The first document will choose the potential cluster, and FGSDMM+ will create a new cluster to store this document. Later documents will choose one of the K_{non} non-empty clusters or the potential cluster with probabilities computed from Equation 25 and Equation 26, respectively. Each time a document chooses the potential cluster, FGSDMM+ will create a new cluster to store that document and decrease the probability of later documents choosing the potential cluster. In this way, the number of non-empty clusters will grow from zero to a certain value, and FGSDMM+ can obtain the initial clustering results. The above initialization approach is identical to the Chinese restaurant process [18] when the maximum number of tables is assumed to be K_{max}.

After initialization, FGSDMM+ will run a Gibbs sampling algorithm several times to obtain the final clustering result. The Gibbs sampling step of FGSDMM+ is similar to FGSDMM, except that FGSDMM+ needs to create a new cluster to store a document when it chooses the potential cluster. The space complexity of FGSDMM+ is linear to the number of non-empty clusters K_{non}, because it only needs to store the information of the non-empty clusters. Besides, FGSDMM+ will introduce less noise into clustering, because it assigns each document to a cluster according to the probability of the document belonging to each cluster.

5. EXPERIMENTAL STUDY

5.1 Experimental Setup

5.1.1 Data Sets

We use three real text datasets in the experimental study:

- **NG20**[1] . This dataset consists of 18,846 documents from 20 major newsgroups which is a classical dataset for the evaluation of text clustering methods. The average length of the documents in NG20 is 137.85.
- **R52**. Similar to [6], we removed the documents with more than one class and the classes with no documents from the 100 most frequent classes in Reuters-21578[2] and obtained a dataset consists of 9,100 documents from 52 classes.
- **Tweet89**[3]. This dataset consists of 2,472 tweets that are highly relevant to 89 queries. The relevance between tweets and queries are manually labelled in the 2011 and 2012 microblog tracks at the Text REtrieval Conference [4] . The average length of the tweets in this dataset is 8.56.

[1]http://qwone.com/~jason/20Newsgroups/
[2]http://www.daviddlewis.com/resources/testcollections/reuters21578/
[3]https://github.com/jackyin12/GSDMM/
[4]http://trec.nist.gov/data/microblog.html

Dataset	D	K	V	Avg Len
NG20	18,846	20	181,754	137.85
R52	9,100	52	12,608	59.61
Tweet89	2,472	89	5,098	8.56

Table 2: Statistics of the text datasets (D:Number of documents, K:Number of clusters, V: Vocabulary size, Avg Len: Average length of the documents)

The preprocessing step includes converting all letters into lowercase, removing stop words, and stemming. After preprocessing, the statistics of these text datasets are shown in Table 2. We can see the average length of the documents in NG20 and R52 is much larger than that of Tweet89. We plan to evaluate the performance of the clustering methods on both short and long texts with these datasets.

5.1.2 Evaluation Metrics

In this part, we introduce the evaluation metrics used in this paper: Homogeneity (H), completeness (C), and Normalized Mutual Information (NMI). We used the implementation of these metrics in sklearn[5] in the experimental study.

Homogeneity and completeness [20] are two intuitive evaluation metric using conditional entropy analysis. Homogeneity represents the objective that each cluster contains only members of a ground true group. Completeness represents the objective that all members of a ground true group are assigned to the same cluster. Homogeneity (H) and completeness (C) scores are formally defined as follows [20]:

$$H = 1 - \frac{\sum_{c,k} n_{c,k} \log\left(\frac{n_{c,k}}{n_k}\right)}{\sum_c n_c \log \frac{n_c}{N}} \tag{27}$$

$$C = 1 - \frac{\sum_{c,k} n_{c,k} \log\left(\frac{n_{c,k}}{n_c}\right)}{\sum_k n_k \log \frac{n_k}{N}} \tag{28}$$

where n_c is the number of documents in class c, n_k is the number of documents in cluster k, $n_{c,k}$ is the number of documents in class c as well as in cluster k, and N is the number of documents in the dataset.

The Normalized Mutual Information (NMI) is widely used to evaluate the quality of the clustering results. NMI measures the amount of statistical information shared by the random variables representing the cluster assignments and the ground true groups of the documents. Normalized Mutual Information (NMI) is formally defined as follows [23]:

$$NMI = \frac{\sum_{c,k} n_{c,k} \log\left(\frac{N \cdot n_{c,k}}{n_c \cdot n_k}\right)}{\sqrt{\left(\sum_c n_c \log \frac{n_c}{N}\right)\left(\sum_k n_k \log \frac{n_k}{N}\right)}} \tag{29}$$

where n_c is the number of documents in class c, n_k is the number of documents in cluster k, $n_{c,k}$ is the number of documents in class c as well as in cluster k, and N is the number of documents in the dataset. When the clustering results perfectly match the ground true classes, the NMI value will be one. While when the clustering results are randomly generated, the NMI value will be close to zero.

5.1.3 Methods for Comparison

We compare FGSDMM+ with the following three clustering methods:

- **K-means**. K-means [10] is probably the most widely used method for clustering. Following [7], we set the similarity metric as cosine similarity. To cope with the problem of falling into local maximum, we set the number of initializations at 10 for each run of K-means.
- **LDA**. We treat the topics found by LDA [5] as clusters and assign each document to the cluster with the highest value in its topic proportion vector. Following [9], we set $\alpha = K/50$ and $\beta = 0.1$ where K is the number of topics assumed by LDA.
- **GSDMM**[3]. This is a state-of-the-art clustering method for short text clustering which is actually the collapsed Gibbs sampling algorithm for the Dirichlet multinomial mixture model. Following [27], we set $\alpha = 0.1$ and $\beta = 0.1$ for GSDMM.

[5]http://www.scikit-learn.org

5.2 Comparison with Existing Methods

In this part, we compare the performance of FGSDMM+ with GSDMM [27], K-means [7], and LDA [5]. We will not report the result of FGSDMM here, because the performance of FGSDMM is the same as GSDMM. All algorithms were implemented in java and conducted on a Linux server with Intel Xeon E5310 1.60GHz CPU and 19GB memory. We vary the number of clusters for each dataset. For FGSDMM+ and GSDMM, they correspond to the assumed maximum number of clusters. For all algorithms, we set the maximum number of iterations at 100 (to make a fair comparison). For each algorithm, we run 20 independent trials on each dataset, and report the mean and standard deviation of the NMI of the results in Table 3.

From Table 3, we can see that FGSDMM+ always achieves the highest performance compared with the other three clustering methods on all datasets. Meanwhile, the standard deviations of the 20 independent trials of FGSDMM+ are quite small which means that FGSDMM+ has high consistency. An interesting observation is that all methods perform better on the short text dataset (Tweet89) than other two long text datasets (NG20 and R52). One possible explanation is that Tweet89 is easier for clustering because it is about events and has smaller vocabulary as shown in Table 2.

5.3 Speed of the Algorithms

In this part, we try to investigate the influence of the assumed maximum number of clusters K_{max} to the speed of FGSDMM+, FGSDMM, and GSDMM. We set $\alpha = 1$, $\beta = 0.05$, and the number of iterations at 30 for all datasets.

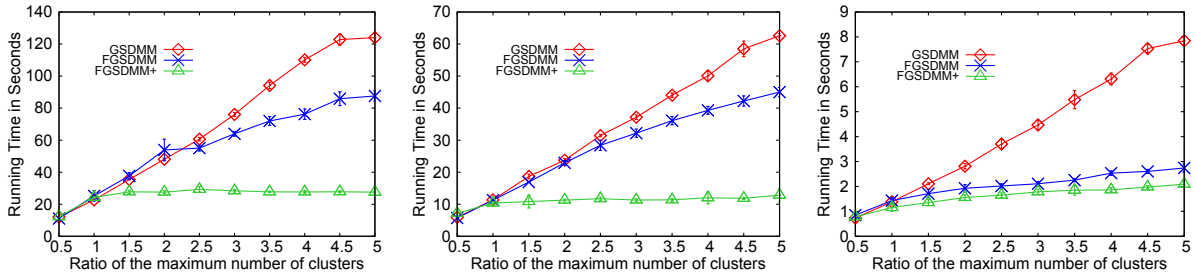
Figure 2 shows the speed of FGSDMM+, FGSDMM, and GSDMM with different assumed maximum number of clusters K_{max}. We set the number of iterations at 30, and vary the assumed maximum number of clusters K_{max} as different times of the true number of clusters in each dataset. We can see that the speed of GSDMM is approximately linear to the assumed maximum number of clusters K_{max}. Because GSDMM needs to re-assign each document to one of the K_{max} clusters according to Equation 25 in each iteration. FGSDMM can improve the speed of GSDMM, however, the improvement is not obvious on NG20 and R52. The speed of FGSDMM+ gets almost stable when the assumed maximum number of clusters is larger than the true number of clusters.

5.4 Influence of the Number of Iterations

In this part, we try to investigate the influence of the number of iterations to the number of clusters found by FGSD-

	NG20			R52			Tweet89		
K	10	20	40	26	52	104	45	89	178
FGSDMM+	**.619** ± .007	**.662** ± .006	**.667** ± .009	**.586** ± .008	**.587** ± .005	**.587** ± .009	**.838** ± .012	**.860** ± .007	**.872** ± .008
GSDMM	.614 ± .006	.658 ± .004	.663 ± .007	.577 ± .004	.577 ± .006	.583 ± .007	.834 ± .005	.853 ± .008	.865 ± .004
K-means	.235 ± .008	.321 ± .006	.352 ± .005	.333 ± .002	.360 ± .003	.383 ± .005	.692 ± .008	.753 ± .006	.745 ± .005
LDA	.584 ± .014	.605 ± .012	.614 ± .012	.493 ± .011	.498 ± .009	.509 ± .012	.773 ± .012	.798 ± .012	.813 ± .013

Table 3: NMI results of the clustering methods.



(a) NG20 (b) R52 (c) Tweet89

Figure 2: Running time of FGSDMM+, FGSDMM, and GSDMM on the three datasets.

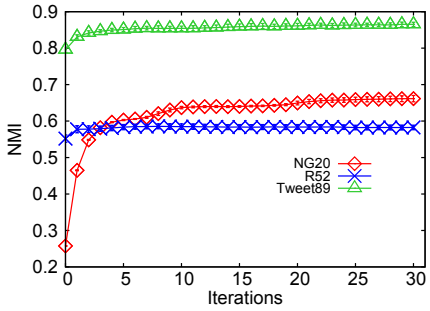


Figure 3: Performance of FGSDMM+ with different number of iterations.

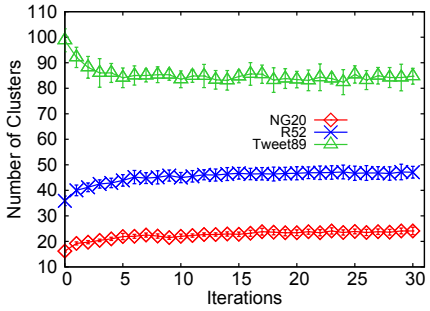


Figure 4: Number of clusters found by FGSDMM+ with different number of iterations.

MM+ and its performance. We set $\alpha = 1$, $\beta = 0.05$, and $K_{max} = 2 * K_{true}$ (K_{true} is the true number of clusters) for all datasets.

Figure 3 shows the performance of FGSDMM+ with different number of iterations. From Figure 3, we can see that FGSDMM+ can get stable performance within about ten iterations. This shows that FGSDMM+ is fast to converge. Another observation is that FGSDMM+ can achieve quite good performance with only the initialization step on R52 and Tweet89. The initialization approach of FGSDMM+ is actually an online clustering algorithm which has two properties: 1) it can process the documents one by one in the order of arriving; 2) it can detect new clusters which means it can deal with the problem of concept drift. While the performance of the initialization step of FGSDMM+ on NG20 is relatively worse, we will try to investigate this phenomenon in our future study on stream text clustering.

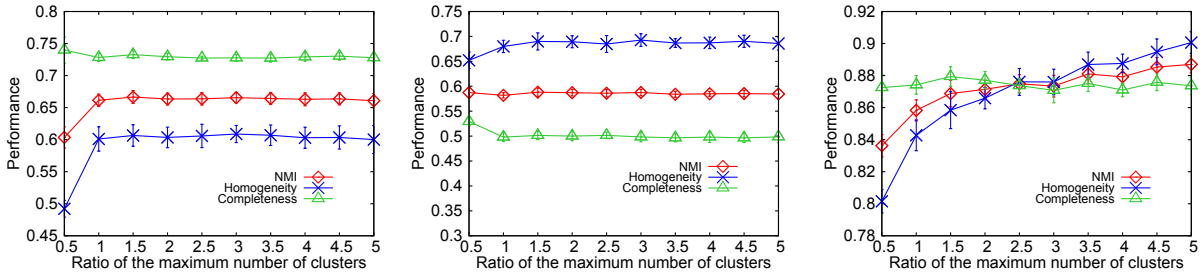
Figure 4 shows the number of clusters found by FGSDMM+ with different number of iterations. From Figure 4, we can see that the number of clusters found by the initialization step FGSDMM+ is near the true number of clusters in the datasets. We can also see that the number of clusters found by FGSDMM+ grows on NG20 and R52, while drops on Tweet89. This means later iterations of FGSDMM+ can amend the result of the initialization step.

5.5 Influence of the Assumed Maximum Number of Clusters

In this part, we try to investigate the influence of the assumed maximum number of clusters K_{max} to the number of clusters found by FGSDMM+ and its performance. We set $\alpha = 1$, $\beta = 0.05$, and the number of iterations at 30 for all datasets.

Figure 5 shows the performance of FGSDMM+ with different assumed maximum number of clusters. We vary the assumed maximum number of clusters as different times of the true number of clusters. From Figure 5, we can see that FGSDMM+ can achieve good performance as long as the assumed maximum number of clusters K_{max} is larger than the true number of clusters in the datasets.

Figure 6 shows the number of clusters found by FGSDMM+ with different assumed maximum number of clusters. We vary the assumed maximum number of clusters as different times of the true number of clusters. Different from NG20 and R52, the number of clusters found on Tweet89 grows slightly when K_{max} gets larger than the true number of clusters. One possible explanation is that the average length of the documents in Tweet89 is only 8.56, which is quite short compared with that of NG20 (137.85) and R52 (59.61). The influence of the similarity between the document and the cluster is relatively small (product of less



(a) NG20 (b) R52 (c) Tweet89

Figure 5: Performance of FGSDMM+ with different maximum number of clusters.

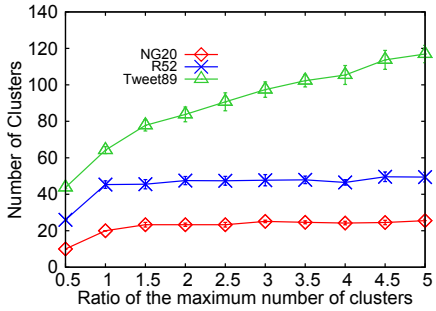


Figure 6: Number of clusters found by FGSDMM+ with different maximum number of clusters.

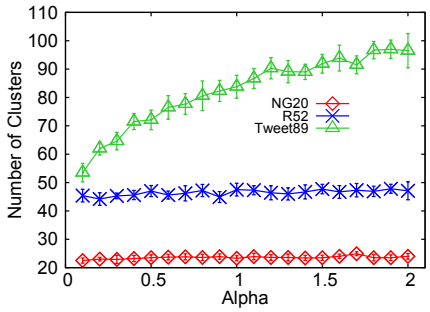


Figure 7: Number of clusters found by FGSDMM+ with different values of α.

than nine fractions averagely) on Tweet89, and the "richer gets richer" property becomes relatively more important. When K_{max} gets larger, the probability of a document choosing the potential cluster gets larger, because there are $(K_{max} - K_{non})\alpha$ pseudo documents in the potential cluster.

We also find that FGSDMM+ can achieve good performance when the assumed maximum number of clusters K_{max} is set to be the number of documents on all the datasets. The number of clusters found on NG20, R52, and Tweet89 are around 26, 55, and 192, respectively.

5.6 Influence of Alpha

In this part, we try to investigate the influence of α to the number of clusters found by FGSDMM+ and its performance. We set $\beta = 0.05$, $K_{max} = 2 * K_{true}$ (K_{true} is the true number of clusters), and the number of iterations at 30 for all datasets.

Figure 7 and Figure 8 show the number of clusters found by FGSDMM+ and the performance of FGSDMM+ with different values of α, respectively. An observation is that the influence of α is similar to the assumed maximum number of clusters K_{max}. The reason is that α is the pseudo number of documents in each cluster, and the probability of the document choosing the potential cluster grows when α gets larger. As the average length of the documents in Tweet89 is only 8.56, the similarity between the document and the cluster plays a relatively small rule, and the growth of α can increase the probability of the documents choosing the potential cluster. On the other hand, the number of clusters found on NG20 and R52 does not grow when we vary α from 0.1 to 2, because the similarity between the document and the cluster plays a so important rule and the increase of α cannot effect the choice of the documents.

5.7 Influence of Beta

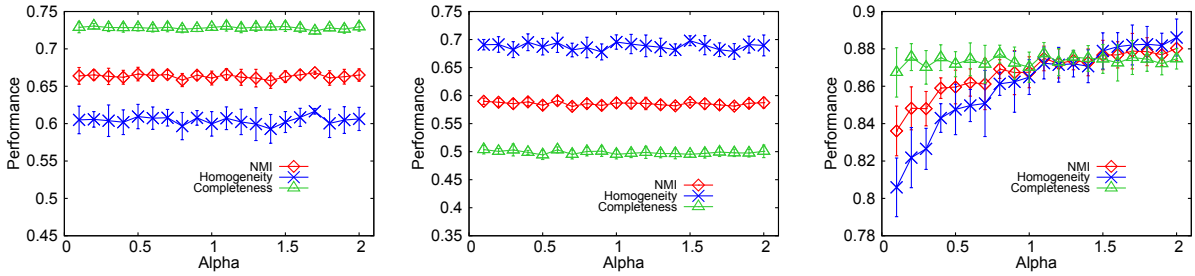
In this part, we try to investigate the influence of β to the number of clusters found by FGSDMM+ and its performance. We set $\alpha = 1$, $K_{max} = 2 * K_{true}$ (K_{true} is the true number of clusters), and the number of iterations at 30 for all datasets.

Figure 9 and Figure 10 show the performance of FGSDMM+ and the number of clusters found by FGSDMM+ with different values of β, respectively. From Figure 10, we can see that the number of clusters found by FGSDMM+ drops when β gets larger. As we know, β is the pseudo frequency of each word in each cluster. When β gets larger, the probability of a document choosing a cluster is less sensitive to the similarity between the document and the cluster. As a result, the influence of the "richer gets richer" property becomes more important, and FGSDMM+ will find fewer clusters. Although the number of clusters found by FGSDMM+ vary a lot when we vary β from 0.01 to 0.1, the NMI of the results remains relatively high. An explanation is that FGSDMM+ can achieve larger completeness with larger β, and can achieve larger homogeneity with smaller β. In other words, FGSDMM+ can balance completeness and homogeneity with β.

6. RELATED WORK

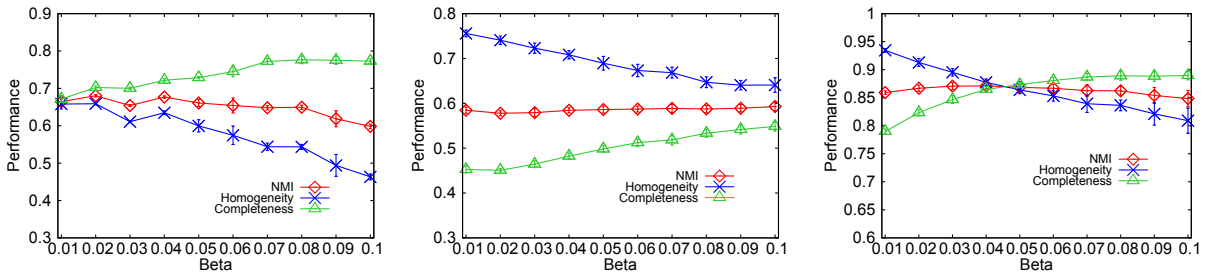
General surveys on text clustering can be found in [1], [2], and [3]. [22] and [28] give experimental comparisons of different text clustering algorithms.

Partitional algorithms like K-means [10] and K-medoids [17] find the clusters by partitioning the entire dataset into a pre-determined number of clusters. The advantage of partitional algorithms is that they are efficient and easy to implement. The Spherical K-means algorithm [7] is used extensively for document clustering due to its low computational and memory requirements. The disadvantages of par-



(a) NG20 (b) R52 (c) Tweet89

Figure 8: Performance of FGSDMM+ with different values of α.



(a) NG20 (b) R52 (c) Tweet89

Figure 9: Performance of FGSDMM+ with different values of β.

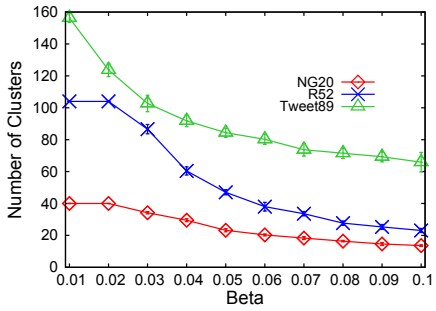


Figure 10: Number of clusters found by FGSDMM+ with different values of β.

titional algorithms is that they need to specify the number of clusters in advance, and they are sensitive to initialization.

Hierarchical algorithms [26] recursively find nested clusters either in an agglomerative mode or a divisive mode. The hierarchical algorithms are particularly useful to support a variety of searching methods because they naturally create a tree-like hierarchy which can be leveraged for the search process [25]. The drawback of hierarchical algorithms is that they need to assume the true number of clusters or a similarity threshold, and their time complexity is quadratic in the number of documents.

Density-based algorithms [8] define the clusters as areas of higher density than the remainder of the dataset. The advantage of density-based algorithms is that they do not need to specify the number of clusters in advance, and can detect the outliers of the dataset. However, they have limitations in handling high-dimensional data like text. Because the feature space of high-dimensional data is usually sparse, density-based algorithms have difficulty to distinguish high-density regions from low-density regions [13].

A comparative study on model-based text clustering algorithms can be found in [29]. The most widely used model-based clustering method is the Gaussian Mixture Model (GMM) [19], which assumes that data points are generated by a mixture of Gaussian distributions. However, the complexity of GMM is too large for high-dimensional data like text. Nigam et al. [16] proposed an EM-based algorithm for the Dirichlet Multinomial Mixture model (DMM) for classification with both labeled and unlabeled documents. When only unlabeled documents are provided, it turns out to be a clustering algorithm.

In [27], we introduced a collapsed Gibbs Sampling algorithm for the DMM model (GSDMM) for text clustering, which can infer the number of clusters automatically as long as the assumed maximum number of clusters is larger than the true number of clusters. GSDMM assumes that there are at most K_{max} clusters in the corpus. In [27], we found that GSDMM can achieve good performance as long as the assumed maximum number of clusters K_{max} is larger than the true number of clusters. However, it is difficult to set a proper maximum number of clusters for GSDMM as we do not know the true number. As a result, we may have to choose a really large K_{max} to ensure safety which will result in the complexity of GSDMM to be large. In addition, GSDMM randomly assigns the documents to K_{max} clusters during initialization, which will introduce noise into clustering. Because the documents in an initial cluster have high probability of coming from different classes.

Topic models like LDA [5] and PLSA [12] are probabilistic generative models that can model texts and identify latent semantics underlying the text collection. In [14], the authors investigated the performance of LDA and PLSA on text clustering. They treat the topics found by topic models as clusters and assign each document to the cluster with the highest value in its topic proportion vector. Different from LDA, we assume that each document is generated by only one topic (cluster) and the words in the document are gen-

erated independently when the document's cluster assignment is known. We find that this model is more effective for the text clustering task, and our extensive experimental study shows that our algorithm can achieve significantly better performance than LDA on both long and short text datasets.

7. CONCLUSION

In this paper, we propose a text clustering algorithm using an online clustering scheme for initialization called FGSDMM+, which has low time and space complexity and can detect the number of clusters automatically. Our extensive experimental study shows that FGSDMM+ can achieve better performance than three other clustering methods on both short and long text datasets. We compared the speed of FGSDMM+, FGSDMM, and GSDMM with different assumed maximum number of clusters K_{max}. We found that the speed of GSDMM is approximately linear to the assumed maximum number of clusters K_{max}, and the speed of FGSDMM+ gets almost stable when the assumed maximum number of clusters is larger than the true number of clusters.

Acknowledgment

This work was supported in part by National Basic Research Program of China (973 Program) under Grant No. 2014CB340505, National Natural Science Foundation of China under Grant No. 61532010 and 61272088, and Tsinghua University Initiative Scientific Research Program under Grant No.20131089256.

8. REFERENCES

[1] C. C. Aggarwal and C. Zhai. A survey of text clustering algorithms. In *Mining Text Data*, pages 77–128. Springer, 2012.

[2] D. C. Anastasiu, A. Tagarelli, and G. Karypis. Document clustering: The next frontier. Technical report, University of Minnesota, 2013.

[3] N. O. Andrews and E. A. Fox. Recent developments in document clustering. Technical report, Computer Science, Virginia Tech, 2007.

[4] C. M. Bishop and N. M. Nasrabadi. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 2003.

[6] A. Cardoso-Cachopo. Improving Methods for Single-label Text Categorization. PdD Thesis, Instituto Superior Tecnico, Universidade Tecnica de Lisboa, 2007.

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

[8] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *SIGKDD*, pages 226–231, 1996.

[9] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235, 2004.

[10] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Applied statistics*, pages 100–108, 1979.

[11] G. Heinrich. Parameter estimation for text analysis. Technical report, 2009.

[12] T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, pages 50–57, 1999.

[13] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

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

[15] G. McLachlan and K. Basford. *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, New York, 1988.

[16] K. Nigam, A. McCallum, S. Thrun, and T. M. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39(2/3):103–134, 2000.

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

[18] J. Pitman. Exchangeable and partially exchangeable random partitions. *Probability Theory and Related Fields*, 102:145–158, 1995.

[19] D. Reynolds. Gaussian mixture models. *Encyclopedia of Biometrics*, pages 659–663, 2009.

[20] A. Rosenberg and J. Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *EMNLP-CoNLL*, pages 410–420, 2007.

[21] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

[22] M. Steinbach, G. Karypis, V. Kumar, et al. A comparison of document clustering techniques. In *KDD workshop on text mining*, volume 400, pages 525–526. Boston, 2000.

[23] A. Strehl and J. Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *The Journal of Machine Learning Research*, 3:583–617, 2003.

[24] Y. W. Teh. Dirichlet process. In *Encyclopedia of machine learning*, pages 280–287. Springer, 2010.

[25] E. M. Voorhees. Implementing agglomerative hierarchic clustering algorithms for use in document retrieval. *Information Processing and Management*, 22(6):465–476, 1986.

[26] P. Willett. Recent trends in hierarchic document clustering: a critical review. *Information Processing and Management*, 24(5):577–597, 1988.

[27] J. Yin and J. Wang. A dirichlet multinomial mixture model-based approach for short text clustering. In *SIGKDD*, pages 233–242, 2014.

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

[29] S. Zhong and J. Ghosh. Generative model-based document clustering: a comparative study. *Knowledge and Information Systems*, 8(3):374–384, 2005.