A Dirichlet Multinomial Mixture Model-based Approach for Short Text Clustering

Jianhua Yin Jianyong Wang
Tsinghua National Laboratory for Information Science and Technology (TNList),
Department of Computer Science and Technology, Tsinghua University, Beijing, China
jhyin12@gmail.com, jianyong@tsinghua.edu.cn

ABSTRACT

Short text clustering has become an increasingly important task with the popularity of social media like Twitter, Google+, and Facebook. It is a challenging problem due to its sparse, high-dimensional, and large-volume characteristics. In this paper, we proposed a collapsed Gibbs Sampling algorithm for the Dirichlet Multinomial Mixture model for short text clustering (abbr. to GSDMM). We found that GSDMM can infer the number of clusters automatically with a good balance between the completeness and homogeneity of the clustering results, and is fast to converge. GSDMM can also cope with the sparse and high-dimensional problem of short texts, and can obtain the representative words of each cluster. Our extensive experimental study shows that GSDMM can achieve significantly better performance than three other clustering models.

Categories and Subject Descriptors

I.5.3 [**PATTERN RECOGNITION**]: Clustering—*Algorithms*

Keywords

Short text clustering; Dirichlet Multinomial Mixture; Gibbs Sampling

1. INTRODUCTION

Berkhin [4] discussed several important issues of clustering: 1) Setting of the number of clusters; 2) Ability to work with high-dimensional data; 3) Interpretability of results; 4) Scalability to large datasets. Short text clustering has all the above challenges. Different from the normal text clustering, short text clustering also has the problem of sparsity [1]. Most words only occur once in each short text, as a result, the Term Frequency-Inverse Document Frequency (TF-IDF) measure cannot work well in the short text setting. Furthermore, if we use the Vector Space Model [25] to represent the short texts, the sparse and high-dimensional vectors will result in waste of both memory and computation time.

In this paper, we try to cope with the above challenges of short text clustering. Specifically, we proposed a collapsed Gibbs Sampling algorithm for the Dirichlet Multinomial Mixture model [20] for short text clustering (abbr. to GSDMM). We also proposed the Movie Group Process (MGP) as an analogy of GSDMM. We can imagine the documents as students in a movie discussion course, and the words of a document as the movies the student has watched. The short text clustering problem turns out to be clustering the students into groups so that students in the same group will share similar interests (similar movie lists), while students in different groups will share different interests. The intuition of MGP is that we can randomly assign the students to K tables, then we ask each student to re-choose a table in turn with two rules: 1) Choose a table with more students; 2) Choose a table whose students share similar interests with him. As this process goes on, some tables will grow larger and others will vanish. Finally, only a part of the tables will still have students and the students in each table will share similar interests. In other words, we can cluster the students into several groups in this way. With the help of MGP, we explored how and why GSDMM works as well as the meaning of its parameters.

We find that GSDMM has the following nice properties: 1) GSDMM can infer the number of clusters automatically; 2) GSDMM has a clear way to balance the completeness and homogeneity of the clustering results; 3) GSDMM is fast to converge; 4) Unlike the Vector Space Model (VSM)-based approaches, GSDMM can cope with the sparse and high-dimensional problem of short texts; 5) Like Topic Models (e.g., PLSA [10] and LDA [6]), GSDMM can also obtain the representative words of each cluster.

In the experimental study, we compared GSDMM with K-means [13], the Hierarchical Agglomerative Clustering (HAC) model [15], and DMAFP [11]. We did not choose other clustering methods like Gaussian Mixture Model [5], Affinity Propagation [8], and Spectral clustering [18], because they are not scalable with high-dimensional and large-volume data like texts. We conducted experiments on four datasets and evaluated the results with five metrics, and found that GSDMM can achieve significantly better performance than K-means, HAC, and DMAFP.

The contributions of this paper are summarized as follows.

- To the best of our knowledge, this is the first attempt to apply the Dirichlet Multinomial Mixture (DMM) model for short text clustering, and our experimental

KDD'14, August 24–27, 2014, New York, NY, USA.
Copyright 2014 ACM 978-1-4503-2956-9/14/08 ...$15.00.
http://dx.doi.org/10.1145/2623330.2623715 .

study has validated its effectiveness. We find it can cope with the sparse and high-dimensional problem of short texts, and can obtain the representative words of each cluster.

- We proposed a collapsed Gibbs Sampling algorithm for DMM (abbr. to GSDMM) which can achieve very good performance on short text clustering. Meanwhile, GSDMM can infer the number of clusters automatically with a good balance between the completeness and homogeneity of the clustering results, and is fast to converge.

- We proposed the Movie Group Process (MGP) as an analogy of GSDMM which can help us understand how and why GSDMM works as well as the meaning of its parameters.

The remainder of this paper is organized as follows. In Section 2, we first propose the Movie Group Process (MGP), then we introduce the DMM model and our GSDMM algorithm. We discuss several important aspects of GSDMM in Section 3. Section 4 describes the design of experiments to evaluate the performance of our algorithm. In Section 5, we review the related work of short text clustering. We finally present conclusions and future work in Section 6.

2. APPROACH

2.1 Movie Group Process

In this part, we introduce an analogy for the short text clustering problem. We will use this analogy in the whole paper, and it can help us understand both the short text clustering problem and our GSDMM algorithm.

We can imagine that the professor of a movie discussion course plans to divide the students into several groups. She expects the students in the same group have watched more movies of the same, so they would have more things to discuss. The professor asks the students to write down the movies they have watched within several minutes. (The list will not be too long because the students do not have enough time, and they will more likely write down movies they watched recently or movies they love much.) Now, each student can be represented with a list of movies. The professor needs to find a way to cluster the students into several groups. The goal is that students in the same group will share similar interests (similar movie lists), while students in different groups will share different interests.

Let us restate the Movie Group Problem more formally and point out its relationship with the short text clustering problem. The input is D students (documents) and each student (document) is represented by a short list of movies (words). The goal is to cluster the students (documents) into several groups, so that students (documents) in the same group are similar and students (documents) in different groups are dissimilar. We define the number of distinct movies (words) as V. The sparse characteristic of short text means that V is really large (often larger than 10^5), while the average number of words ($\bar{L}$) in each short text is small (often less than 10^2).

We first discuss the common similarity-based clustering models for this problem before introducing our approach. Common similarity-based models like K-means [13] and HAC [15] for text clustering usually represent the documents with the Vector Space Model (VSM) [25]. Each document (student) is represented with a vector of length V. Each element of the vector is the weight of the corresponding word (e.g., TF-IDF). Due to the sparse problem of short texts, most words of the documents have TF=1, which means TF is almost useless in the representation of short texts. Although each short document has only a small number of words, it is represented with a vector of size V (often larger than 10^5). This VSM representation of documents results in both high time complexity and high space complexity, and is related to the high-dimensional problem of the short texts.

We can imagine that the professor invites the students into a huge restaurant and randomly assigns the students to K tables. Then she asks the students to re-choose a table in turn. We can expect that a student will choose a table according to the following two rules:

- Rule 1: Choose a table with more students.

- Rule 2: Choose a table whose students share similar interests (i.e., watched more movies of the same) with him.

As this process goes on, some tables will grow larger and others will vanish. We can expect that finally only a part of the tables will still have students and the students in each table will share similar interests. In other words, the professor can cluster the students into several groups in this way.

We call the above process as the Movie Group Process (MGP). We can see that the above two natural rules are related to the two goals of clustering: Completeness and Homogeneity [24]. Completeness represents the objective that all members of a ground true group are assigned to the same cluster. Rule 1 of MGP tends to result in high completeness, as it leads popular tables to be more popular (larger clusters are more likely to get larger), and students in the same ground true group are more likely to be in the same table (cluster). This is also known as the "richer gets richer" property. Homogeneity represents the objective that each cluster contains only members of a single ground true group. Rule 2 of MGP tends to result in high homogeneity, because it leads the students in the same table to be more similar (more likely to be in the same ground true group).

To our surprise, the Movie Group Process (MGP) is equivalent to our collapsed Gibbs Sampling algorithm for the Dirichlet Multinomial Mixture model (abbr. to GSDMM). We will first introduce the Dirichlet Multinomial Mixture model in the next section, then we will introduce our GSDMM algorithm in Section 2.3 and Section 2.4.

2.2 Dirichlet Multinomial Mixture

In this section, we introduce the Dirichlet Multinomial Mixture (DMM) model used in Nigam et al. [20]. DMM is a probabilistic generative model for documents, and embodies two assumptions about the generative process: (1) the documents are generated by a mixture model [17], and (2) there is a one-to-one correspondence between mixture components and clusters. When generating document d, DMM first selects a mixture component (cluster) k according to the mixture weights (weights of clusters), $p(z = k)$. Then document d is generated by the selected mixture component (cluster) from distribution $p(d|z = k)$. Thus we can characterize the likelihood of document d with the sum of the total

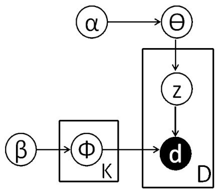

Figure 1: Graphical model of DMM.

V	number of words in the vocabulary
D	number of documents in the corpus
$\bar{L}$	average length of documents
$\vec{d}$	documents in the corpus
$\vec{z}$	cluster labels of each document
I	number of iterations
m_z	number of documents in cluster z
n_z	number of words in cluster z
n_z^w	number of occurrences of word w in cluster z
N_d	number of words in document d
N_d^w	number of occurrences of word w in document d

Table 1: Notations

probability over all mixture components:

$$p(d) = \sum_{k=1}^{K} p(d|z=k)p(z=k) \tag{1}$$

Here, K is the number of mixture components (clusters). Now, the problem becomes how to define $p(d|z=k)$ and $p(z=k)$. DMM makes the Naive Bayes assumption: that the words in a document are generated independently when the document's cluster label k is known, and the probability of a word is independent of its position within the document. Then the probability of document d generated by cluster k can be derived as follows:

$$p(d|z=k) = \prod_{w \in d} p(w|z=k) \tag{2}$$

Nigam et al. [20] assumes that each mixture component (cluster) is a multinomial distribution over words, such that $p(w|z=k) = p(w|z=k,\Phi) = \phi_{k,w}$, where $w = 1,...,V$ and $\sum_w \phi_{k,w} = 1$. They assume a Dirichlet distribution as the prior for each mixture component (cluster), such that $p(\Phi|\vec{\beta}) = Dir(\vec{\phi}_k|\vec{\beta})$. They also assume that the weight of each mixture component (cluster) is sampled from a multinomial distribution, such that $p(z=k) = p(z=k|\Theta) = \theta_k$, where $k = 1,...,K$ and $\sum_k \theta_k = 1$. In addition, they assume a Dirichlet prior for this multinomial distribution, such that $p(\Theta|\vec{\alpha}) = Dir(\vec{\theta}|\vec{\alpha})$.

The graphical model of DMM is shown in Figure 1. In our short text clustering problem, we need to estimate the mixture component (cluster) z for each document d. We will introduce our GSDMM algorithm with the help of the Movie Group Process (MGP) in the next section.

2.3 Gibbs Sampling for DMM

In this section, we introduce the collapsed Gibbs Sampling algorithm for the Dirichlet Multinomial Mixture model (abbr. to GSDMM), which is equivalent to the Movie Group Process (MGP) introduced in Section 2.1.

The detail of our GSDMM algorithm is shown in Algorithm 1, and the meaning of its variables is shown in Table 1. In the initialization step, we randomly assign the documents to K clusters, and record the following information: $\vec{z}$ (cluster labels of each document), m_z (number of documents in cluster z), n_z (number of words in cluster z), and n_z^w (number of occurrences of word w in cluster z). Then we traverse the documents for I iterations. (In Section 4.4, we found that GSDMM can achieve good and stable performance when I equals five.) In each iteration, we re-assign a cluster for each document d in turn according to the conditional distribution: $p(z_d = z|\vec{z}_{\neg d}, \vec{d})$, where $\neg d$ means the cluster label of document d is removed from $\vec{z}$. Each time we re-assign a cluster z to document d, the corresponding information in $\vec{z}$, m_z, n_z, and n_z^w are updated accordingly. Finally, only a part of the initial K clusters will remain non-empty, in other words, GSDMM can cluster the documents into several groups. Through experimental study in Section 4.5, we found that the number of non-empty clusters found by GSDMM can be near the true number of groups as long as K is larger than the true number. GSDMM is also a soft clustering model like Gaussian Mixture Model (GMM) [5], since we can get the probability of each document belonging to each cluster from $p(z_d = z|\vec{z}_{\neg d}, \vec{d})$.

Algorithm 1: GSDMM

```
Data: Documents in the input, $\vec{d}$.
Result: Cluster labels of each document, $\vec{z}$.
begin
    initialize $m_z, n_z$, and $n_z^w$ as zero for each cluster $z$
    for each document $d \in [1, D]$ do
        sample a cluster for $d$:
        $z_d \leftarrow z \sim Multinomial(1/K)$
        $m_z \leftarrow m_z + 1$ and $n_z \leftarrow n_z + N_d$
        for each word $w \in d$ do
            $n_z^w \leftarrow n_z^w + N_d^w$
    for $i \in [1, I]$ do
        for each document $d \in [1, D]$ do
            record the current cluster of $d$: $z = z_d$
            $m_z \leftarrow m_z - 1$ and $n_z \leftarrow n_z - N_d$
            for each word $w \in d$ do
                $n_z^w \leftarrow n_z^w - N_d^w$
            sample a cluster for $d$:
            $z_d \leftarrow z \sim p(z_d = z|\vec{z}_{\neg d}, \vec{d})$ (Equation 4)
            $m_z \leftarrow m_z + 1$ and $n_z \leftarrow n_z + N_d$
            for each word $w \in d$ do
                $n_z^w \leftarrow n_z^w + N_d^w$
```

We can derive $p(z_d = z|\vec{z}_{\neg d}, \vec{d})$ from the Dirichlet Multinomial Mixture (DMM) model, and find that it conforms to the two rules of MGP introduced in Section 2.1. We just introduce the results directly here, and will explain the derivation details in the next section.

If we assume each word can at most appear once in each document (In the movie group example, the assumption is that a movie can at most appear once in each student's list). We can derive a quite elegant form of the conditional distribution as follows:

$$p(z_d = z|\vec{z}_{\neg d}, \vec{d}) \propto \frac{m_{z,\neg d} + \alpha}{D - 1 + K\alpha} \frac{\prod_{w \in d}(n_{z,\neg d}^w + \beta)}{\prod_{i=1}^{N_d}(n_{z,\neg d} + V\beta + i - 1)} \tag{3}$$

where N_d is the number of words in document d. In short text setting, N_d is often less than 100.

The first part of Equation 3 relates to Rule 1 of MGP (Choose a table with more students). Here $m_{z,\neg d}$ is the number of students (documents) in table z without considering student d, and D is the total number of students. When table z has more students, the first part tends to be larger, and a student will tend to choose a table with more students. As a result, the first part of Equation 3 tends to result in large completeness, because it leads large tables (clusters) to be larger and students in the same ground true group are more likely to be in the same table (cluster). The second part of Equation 3 relates to Rule 2 of MGP (Choose a table whose students share similar interests with him). Here $n^w_{z,\neg d}$ and $n_{z,\neg d}$ are the number of occurrences of movie w in table z and the total number of movies in table z without considering student d, respectively. When table z has more students sharing similar interests with student d (i.e., watched more movies of the same), movies of student d will appear more often in table z (with larger $n^w_{z,\neg d}$), and the probability of student d choosing table z will be larger. As a result, the second part of Equation 3 tends to result in large homogeneity, because it can leads the students in the same table to be more similar (more likely to be in the same ground true group).

If we allow a word to appear multi-times in a document (A movie can appear multi-times in a student's list). We can derive the conditional probability as follows:

$$p(z_d = z|\vec{z}_{\neg d}, \vec{d}) \propto \frac{m_{z,\neg d} + \alpha}{D - 1 + K\alpha} \frac{\prod_{w \in d} \prod_{j=1}^{N^w_d} (n^w_{z,\neg d} + \beta + j - 1)}{\prod_{i=1}^{N_d} (n_{z,\neg d} + V\beta + i - 1)} \quad (4)$$

where N^w_d is the number of occurrences of word w in document d. We should note that the two parts of Equation 4 have similar relationship with MGP like that of Equation 3, and the complexity of Equation 4 is the same as Equation 3. The only difference between them is the numerator of their second part. We will try to derive Equation 3 and Equation 4 from the Dirichlet Multinomial Mixture (DMM) model in the next section.

2.4 Derivation of GSDMM

In this section, we try to formally derive the conditional distribution $p(z_d = z|\vec{z}_{\neg d}, \vec{d})$ used in our GSDMM algorithm as follows.

$$p(z_d = z|\vec{z}_{\neg d}, \vec{d}) = \frac{p(\vec{d}, \vec{z}|\vec{\alpha}, \vec{\beta})}{p(\vec{d}, \vec{z}_{\neg d}|\vec{\alpha}, \vec{\beta})} \propto \frac{p(\vec{d}, \vec{z}|\vec{\alpha}, \vec{\beta})}{p(\vec{d}_{\neg d}, \vec{z}_{\neg d}|\vec{\alpha}, \vec{\beta})} \quad (5)$$

where $\neg d$ means document d is excluded from $\vec{z}$ and $\vec{d}$. Now we need to derive the full distribution $p(\vec{d}, \vec{z}|\vec{\alpha}, \vec{\beta})$. From the graphical model of DMM in Figure 1, we can see $p(\vec{d}, \vec{z}|\vec{\alpha}, \vec{\beta}) = p(\vec{d}|\vec{z}, \vec{\beta})p(\vec{z}|\vec{\alpha})$. Then we need to derive $p(\vec{d}|\vec{z}, \vec{\beta})$ and $p(\vec{z}|\vec{\alpha})$.

Let us first investigate how to obtain $p(\vec{z}|\vec{\alpha})$. We can see that $p(\vec{z}|\vec{\alpha})$ can be obtained by integrating with respect to Θ as $p(\vec{z}|\vec{\alpha}) = \int p(\vec{z}|\Theta)p(\Theta|\vec{\alpha})d\Theta$. As mentioned in Section 2.2, $p(\Theta|\vec{\alpha})$ is a Dirichlet distribution and $p(\vec{z}|\Theta)$ is a multinomial distribution. With similar techniques of [9], we can get $p(\vec{z}|\vec{\alpha}) = \frac{\Delta(\vec{m}+\vec{\alpha})}{\Delta(\vec{\alpha})}$, where $\vec{m} = \{m_k\}_{k=1}^K$, and m_k is the number of documents (students) in the kth cluster (table). Here we adopt the Δ function in [9], and we have $\Delta(\vec{\alpha}) = \frac{\prod_{k=1}^K \Gamma(\alpha)}{\Gamma(\sum_{k=1}^K \alpha)}$ and $\Delta(\vec{m} + \vec{\alpha}) = \frac{\prod_{k=1}^K \Gamma(m_k+\alpha)}{\Gamma(\sum_{k=1}^K (m_k+\alpha))} = \frac{\prod_{k=1}^K \Gamma(m_k+\alpha)}{\Gamma(D+K\alpha)}$, where D is the number of documents in the dataset, $D = \sum_{k=1}^K m_k$.

Similarly, $p(\vec{d}|\vec{z}, \vec{\beta})$ can be obtained by integrating with respect to Φ as $p(\vec{d}|\vec{z}, \vec{\beta}) = \int p(\vec{d}|\vec{z}, \Phi)p(\Phi|\vec{\beta})d\Phi = \prod_{k=1}^K \frac{\Delta(\vec{n}_k+\vec{\beta})}{\Delta(\vec{\beta})}$, where $\vec{n}_k = \{n^w_k\}_{w=1}^V$, and n^w_k is the number of occurrences of word w in the kth cluster (table). Similarly, $\Delta(\vec{\beta}) = \frac{\prod_{w=1}^V \Gamma(\beta)}{\Gamma(\sum_{w=1}^V \beta)}$ and $\Delta(\vec{n}_k+\vec{\beta}) = \frac{\prod_{w=1}^V \Gamma(n^w_k+\beta)}{\Gamma(\sum_{w=1}^V (n^w_k+\beta))} = \frac{\prod_{w=1}^V \Gamma(n^w_k+\beta)}{\Gamma(n_k+V\beta)}$, where n_k is number of words (movies) in document (table) k, that is, $n_k = \sum_{w=1}^V n^w_k$.

Now the joint distribution becomes:

$$p(\vec{d}, \vec{z}|\vec{\alpha}, \vec{\beta}) = \frac{\Delta(\vec{m} + \vec{\alpha})}{\Delta(\vec{\alpha})} \prod_{k=1}^K \frac{\Delta(\vec{n}_k + \vec{\beta})}{\Delta(\vec{\beta})}$$

Then the conditional distribution in Equation 5 can be derived as follows:

$$\begin{aligned} p(z_d = z|\vec{z}_{\neg d}, \vec{d}) &\propto \frac{p(\vec{d}, \vec{z}|\vec{\alpha}, \vec{\beta})}{p(\vec{d}_{\neg d}, \vec{z}_{\neg d}|\vec{\alpha}, \vec{\beta})} \\ &\propto \frac{\Delta(\vec{m} + \vec{\alpha})}{\Delta(\vec{m}_{\neg d} + \vec{\alpha})} \frac{\Delta(\vec{n}_z + \vec{\beta})}{\Delta(\vec{n}_{z,\neg d} + \vec{\beta})} \\ &\propto \frac{\Gamma(m_z + \alpha)}{\Gamma(m_{z,\neg d} + \alpha)} \frac{\Gamma(D - 1 + K\alpha)}{\Gamma(D + K\alpha)} \\ &\frac{\prod_{w \in d} \Gamma(n^w_z + \beta)}{\prod_{w \in d} \Gamma(n^w_{z,\neg d} + \beta)} \frac{\Gamma(n_{z,\neg d} + V\beta)}{\Gamma(n_z + V\beta)} \end{aligned} \quad (6)$$

where $m_z = m_{z,\neg d} + 1$ and $n_z = n_{z,\neg d} + N_d$. Because Γ function has the following property: $\frac{\Gamma(x+m)}{\Gamma(x)} = \prod_{i=1}^m (x + i - 1)$. We can rewrite Equation 6 into the following form:

$$\begin{aligned} &p(z_d = z|\vec{z}_{\neg d}, \vec{d}) \\ &\propto \frac{m_{z,\neg d} + \alpha}{D - 1 + K\alpha} \frac{\frac{\prod_{w \in d} \Gamma(n^w_z+\beta)}{\prod_{w \in d} \Gamma(n^w_{z,\neg d}+\beta)}}{\prod_{i=1}^{N_d} (n_{z,\neg d} + V\beta + i - 1)} \end{aligned} \quad (7)$$

When we assume each word can at most appear once in each document (In the movie group example, the assumption is a movie can at most appear once in each student's list). We can get $\frac{\prod_{w \in d} \Gamma(n^w_z+\beta)}{\prod_{w \in d} \Gamma(n^w_{z,\neg d}+\beta)} = \prod_{w \in d}(n^w_{z,\neg d} + \beta)$ since $n^w_z = n^w_{z,\neg d} + 1$ holds, and Equation 7 turns out to be Equation 3.

When we allow a word to appear multi-times in each document (A movie can appear multi-times in each student's list).We can get $\frac{\prod_{w \in d} \Gamma(n^w_z+\beta)}{\prod_{w \in d} \Gamma(n^w_{z,\neg d}+\beta)} = \prod_{w \in d} \prod_{j=1}^{N^w_d} (n^w_{z,\neg d} + \beta + j - 1)$ since $n^w_z = n^w_{z,\neg d} + N^w_d$ holds, and Equation 7 turns out to be Equation 4.

3. DISCUSSION

3.1 Meaning of Alpha and Beta

In this part, we try to explore the meaning of α and β with the help of the Movie Group Process (MGP) as introduced in Section 2.1. From Equation 4, we can see that α relates to the prior probability of a student (document) choosing a table (cluster). If we set $\alpha = 0$, a table will never be chosen by the students once it gets empty, because the first part of Equation 4 is now zero. When α gets larger, the probability of a student choosing an empty table will also gets larger.

We can see β is in the second part of Equation 4 which relates to Rule 2 of MGP (Choose a table whose students share similar interests with him). If we set $\beta = 0$, a student will never choose a table when its movie lists do not contain a movie of the student. We can see this is not reasonable, because other movies of the student may appear many times in that table and he may share many similar interests with the students of that table.

DMM assumes symmetric priors for the Dirichlet distributions, in other words, it gives the same α's for all tables (clusters) and the same β's for all movies (words). The same α's for all tables implies that different tables (clusters) are equally important at start. This is consistent to our intuition. While the same β's for all movies implies that different movies (words) are equally important. This is not reasonable, for example, a movie watched by every student cannot give any positive information for clustering the students into groups, it actually will mislead the clustering algorithm. We should give less emphasis on too popular movies (words that appear in too many documents). Intuitively, we can achieve this goal by giving larger β's for these less important words. This means that GSDMM has potential to incorporate the global weighting metrics (e.g., IDF) for words which is a really important technique for document analysis. We plan to explore this in future.

3.2 Relationship with Naive Bayes Classifier

The conditional distribution $p(z_d = z|\vec{z}_{\neg d}, \vec{d})$ in Equation 3 is equivalent to the Naive Bayes Classifier (NBC) [16]. Intuitively, we can assign document d to a cluster z with the largest conditional probability $p(z_d = z|\vec{z}_{\neg d}, \vec{d})$ in Algorithm 1. However, in GSDMM we choose to sample cluster z from the conditional distribution $p(z_d = z|\vec{z}_{\neg d}, \vec{d})$. GSDMM can avoid falling into a local minimum which is a common problem of EM-based algorithms in this way.

We should note that GSDMM has potentially well performance for incremental clustering. We can first group a number of documents into clusters with GSDMM, and a Naive Bayes Classifier (NBC) is learned during this process. Then each time a new document arrives, we can classify it to one of the clusters with the Naive Bayes Classifier and update the classifier (update $\vec{z}$, m_z, n_z, and n_z^w). We plan to explore this in future.

The conditional distribution $p(z_d = z|\vec{z}_{\neg d}, \vec{d})$ in Equation 4 is equivalent to the Bayesian Naive Bayes Classifier (BNBC) [22]. Rennie [22] points out that BNBC performs worse in classification, because it over-emphasizes words that appear more than once in a test document. For example, if word (movie) w appears twice in a document (student) d, the contribution of w for Equation 3 is $(n_{z,\neg d}+\beta)^2$, while the contribution of w for Equation 4 is $(n_{z,\neg d}+\beta)(n_{z,\neg d}+\beta+1)$. The difference is greater when a word appears more frequently in a document. However, this is a good property in the text clustering problem, because words in a document tend to appear in bursts: if a word appears once, it is more likely to appear again [14],[23]. The conditional distribution $p(z_d = z|\vec{z}_{\neg d}, \vec{d})$ in Equation 4 can give words that appear multi-times in a document more emphasis, and allows GSDMM to capture the burstiness of words [7],[14].

3.3 Representation of Clusters

From Algorithm 1, we can see that GSDMM can assign each document to a cluster. With the fact that the Dirichlet distribution is conjugate to the multinomial distribution, we have:

$$p(\vec{\phi}_z|\vec{d}, \vec{z}, \vec{\beta}) = Dir(\vec{\phi}_z|\vec{n}_z + \vec{\beta}) \tag{8}$$

where $\vec{n}_z = \{n_z^w\}_{w=1}^V$, and n_z^w is the number of occurrences of word w in the zth cluster (table).

From Equation 8, we can obtain the posterior mean of Φ as follows:

$$\phi_{z,w} = \frac{n_z^w + \beta}{\sum_{w=1}^V n_z^w + V\beta} \tag{9}$$

where $\phi_{z,w}$ corresponds to the probability of word w being generated by cluster z, and can be regarded as the importance of word w to cluster z. As a result, GSDMM can obtain the representative words of each cluster like Topic Models (e.g., PLSA [10] and LDA [6]).

3.4 Complexity of GSDMM

In this part, we analyze the time and space complexity of GSDMM with a comparison of K-means [13]. The meaning of the notations is shown in Table 1.

First, we analyze the space complexity of GSDMM and K-means. From Algorithm 1, we can see GSDMM needs to store $\vec{z}$, m_z, n_z, and n_z^w. Their size are D, K, K, and KV, respectively. We can see none of them need much space. Actually, when dealing with huge datasets, GSDMM spends most space to store the words in each document with complexity $O(D\bar{L})$. Where $\bar{L}$ (often less than 10^2 in short text corpus) is the average length of the documents. K-means needs to represent the documents as vectors of size V (often larger than 10^5). Therefore, the space complexity of K-means is $O(DV)$. The vector space representation of short documents wastes much space, because most of the elements of the vectors are zero.

Next, we analyze the time complexity of each iteration of GSDMM and K-means. From Algorithm 1, we can see that for each iteration GSDMM needs to re-sample a cluster for the D documents in turn, and for each document d, it computes the conditional probability $p(z_d = z|\vec{z}_{\neg d}, \vec{d})$ for each cluster z. The complexity of $p(z_d = z|\vec{z}_{\neg d}, \vec{d})$ in Equation 4 is linear to the average length of documents, $\bar{L}$. Therefore, GSDMM's time complexity for each iteration is $O(KD\bar{L})$. The complexity of each iteration of K-means is $O(KDS)$, where S is the maximum number of non-zero elements in the vectors of centroids of the clusters [26]. We should note that the centroids of the clusters are not sparse, and S can be really large (even close to V).

4. EXPERIMENTAL STUDY

4.1 Data Sets

4.1.1 Google News

Similar to [2], we utilize Google News[1] as one of the labeled datasets to evaluate the clustering performance. In the Google News, the news articles are grouped into clusters (stories) automatically. We took a snapshot of the Google News on November 27, 2013, and crawled the titles and snippets of 11,109 news articles belonging to 152 clusters. We manually examined this dataset, and found that it is with really good quality (Almost all articles in the same

[1]http://news.google.com

cluster are about the same event, and articles in different clusters are about different events).

We further divided the dataset into three datasets: TitleSet (TSet), SnippetSet (SSet), and TitleSnippetSet (TSSet). The TitleSet and SnippetSet only contain the titles and snippets, respectively, while the TitleSnippetSet contains both the titles and snippets. We use these three datasets to test the performance of different clustering methods on short texts with different length.

4.1.2 *TweetSet*

In the 2011 and 2012 microblog tracks at Text REtrieval Conference (TREC)[2] , totally 109 queries were used. Using a standard polling strategy, the NIST assessors evaluated the tweets submitted for each query by the participants into: spam, not relevant, relevant, and highly-relevant. We regard the queries as clusters and the highly-relevant tweets of each query as documents in each cluster. After removing the queries with none highly-relevant tweets, we constructed a dataset with 89 clusters and totally 2,472 tweets. We refer this dataset as TweetSet.

4.1.3 *PreProcessing*

For all datasets, we conducted the following preprocessing: (1) Convert letters into lowercase; (2) Remove non-latin characters and stop words; (3) Perform stemming for words with the WordNet Lemmatizer of NLTK[3] ; (4) Remove words whose length are smaller than 2 or larger than 15; (5) Remove words with document frequency less than 2.

4.2 Evaluation Metrics

In this part, we introduce the evaluation metrics used in this paper: Homogeneity (H), Completeness (C), V-Measure, Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), and Adjusted Mutual Information (AMI). We used the implementation of these metrics in sklearn[4] in the experimental study.

Homogeneity, Completeness, and V-Measure are used in [24]. Homogeneity represents the objective that each cluster contains only members of a ground true group and completeness represents the objective that all members of a ground true group are assigned to the same cluster. V-measure is an entropy-based measure which explicitly measures how successfully the criteria of homogeneity and completeness have been satisfied, and is actually equivalent to Normalized Mutual Information (NMI) that we will discuss later [3]. As a result, we will only report NMI in our experimental results.

We want to assign two documents to the same cluster if and only if they are similar, and clustering can be viewed as a series of pair-wise decisions. Rand index measures the percentage of decisions that are correct. Adjusted Rand Index (ARI) is the corrected-for-chance version of Rand index, whose expected value is zero [12].

Normalized Mutual Information (NMI) is often used in literature, which measures the amount of statistical information shared by the random variables representing the cluster assignments and the ground true groups of the documents. However, NMI is not adjusted for chance and will tend to increase as the number of different labels (clusters) increases. Adjusted Mutual Information(AMI) [19] corrects the effect of agreement solely due to chance between clusterings, similar to the way Adjusted Rand Index (ARI) corrects the Rand index.

[2]http://trec.nist.gov/data/microblog.html
[3]http://www.nltk.org
[4]http://scikit-learn.org

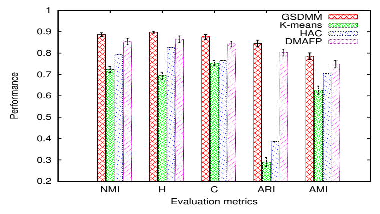

Figure 2: Performance of the models on the TweetSet.

		GSDMM	K-means	DMAFP
TSet	NMI	0.874 ± 0.007	0.732 ± 0.007	0.852 ± 0.009
	H	0.853 ± 0.010	0.692 ± 0.009	0.831 ± 0.010
	C	0.896 ± 0.006	0.775 ± 0.006	0.875 ± 0.007
	ARI	0.693 ± 0.043	0.133 ± 0.030	0.657 ± 0.051
	AMI	0.831 ± 0.012	0.639 ± 0.011	0.814 ± 0.015
SSet	NMI	0.896 ± 0.006	0.759 ± 0.008	0.868 ± 0.008
	H	0.871 ± 0.008	0.754 ± 0.009	0.846 ± 0.011
	C	0.921 ± 0.005	0.764 ± 0.009	0.892 ± 0.007
	ARI	0.746 ± 0.014	0.262 ± 0.017	0.703 ± 0.018
	AMI	0.853 ± 0.009	0.708 ± 0.008	0.819 ± 0.012
TSSet	NMI	0.928 ± 0.004	0.834 ± 0.005	0.901 ± 0.008
	H	0.911 ± 0.005	0.836 ± 0.005	0.889 ± 0.006
	C	0.945 ± 0.003	0.832 ± 0.005	0.912 ± 0.004
	ARI	0.789 ± 0.018	0.370 ± 0.029	0.736 ± 0.023
	AMI	0.897 ± 0.006	0.800 ± 0.006	0.847 ± 0.009

Table 2: Performance of GSDMM, K-means, and DMAFP on TitleSet, SnippetSet, and TitleSnippetSet.

4.3 Comparison of clustering models

In this part, we compare the performance of GSDMM with K-means [13], HAC [15], and DMAFP [11]. K-means and HAC are two popular similarity-based clustering models. DMAFP [11] is a state-of-the-art model-based clustering approach which can infer the number of clusters automatically.

For K-means and HAC, we set K at the true number of clusters of each dataset. For DMAFP and GSDMM, we set the initial number of clusters at 500 for all datasets. For K-means and DMAFP, we set the number of initializations at 20 to cope with their problem of falling into local maximum, and in each run we stop the algorithm when the relative change of the objective function or log marginal probability is less than $1e^{-7}$. For GSDMM, we set the number of iterations at 30, $\alpha = 0.1$, and $\beta = 0.1$ for all datasets. We run each model 20 times on each dataset, and report the mean and standard deviation of their performance measured by the five evaluation metrics introduced in Section 4.2.

Figure 2 shows the performance of GSDMM, K-means, HAC, and DMAFP on the TweetSet. First, we can see that GSDMM performs significantly better than K-means and HAC in terms of all the five metrics. This is because GSDMM can infer the number of clusters automatically with a good balance between the completeness and homogeneity of the clustering results. We will discuss this more deeply in Section 4.5. Second, GSDMM also performs better than DMAFP. This is because DMAFP is an EM-based algorithm which has the problem of falling into a local minimum, while GSDMM does not have the problem of falling into a local minimum as discussed in Section 3.2. We should note that

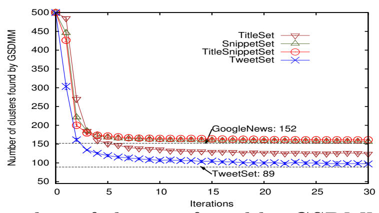

Figure 3: Number of clusters found by GSDMM with different number of iterations on the four datasets.

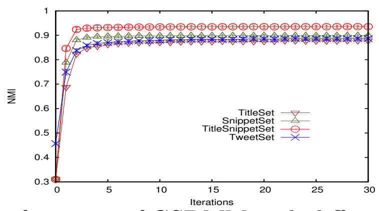

Figure 4: Performance of GSDMM with different number of iterations on the four datasets.

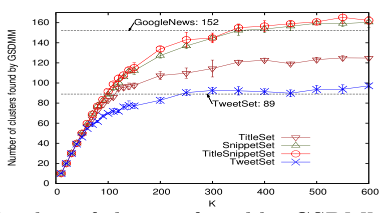

Figure 5: Number of clusters found by GSDMM with different values of K on the four datasets.

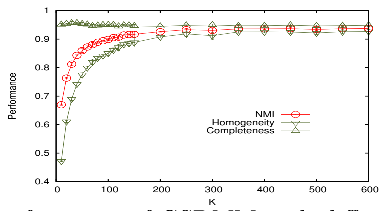

Figure 6: Performance of GSDMM with different values of K on the TitleSnippetSet.

GSDMM is much simpler than DMAFP, and we can understand how and why GSDMM works as well as the meaning of its parameters with the analogy of MGP. Third, GSDMM's ARI is significantly better than that of K-means and HAC. This means GSDMM's performance is much better than the random label assignments, while K-means and HAC are just a little better than the random case.

We find that HAC is not scalable with the Google News datasets. This is because the space complexity of HAC is quadratic in the number of documents. Therefore, we only report the performance of GSDMM, K-means, and DMAFP on the Google News datasets in Table 2. First, we can see GSDMM can achieve better performance than K-means and DMAFP on all the three datasets measured by all the five evaluation metrics. Second, we can see all the three models can achieve better performance on datasets with longer documents, which means we can improve the performance of short text clustering by enriching their representation with methods like [2]. Third, GSDMM can achieve relatively good performance on the TitleSet, even better than the performance of K-means on the TitleSnippetSet. This validates the effectiveness of GSDMM for short text clustering.

4.4 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 GSDMM and the performance of GSDMM. We set the initial number of clusters at 500, $\alpha = 0.1$, and $\beta = 0.1$ for all datasets.

Figure 3 shows the number of clusters found by GSDMM with different number of iterations. From Figure 3, we can see that the number of clusters drops quickly and almost gets stable within about ten iterations. We take the TweetSet as an example to explain this phenomenon. In the initialization step, GSDMM randomly assigns the 2,472 documents (students) of the TweetSet to 500 clusters (tables). For each iteration, GSDMM enumerates the students and lets them re-choose a table according to the two rules of MGP introduced in Section 2.1 in turn. In the beginning, some clusters get large quickly, because once they are chosen by a student, they are more likely to be chosen by other students that share similar interests with them. Similarly, some clusters vanish quickly. We can see that on the TweetSet, after one iteration the number of non-empty clusters drops to about 300 and after two iterations the number of non-empty clusters drops to about 160. With more iterations the allocation of documents will get stable.

Figure 4 shows the performance of GSDMM with different number of iterations measured by NMI on the four datasets. From Figure 4, we can see only after two iterations, GSDMM can achieve quite good performance, and can reach stable performance with about five iterations on all the four datasets. GSDMM can update the model every time it reassigns a document to a cluster which means it can update the model D times in each iteration (D is the number of documents in the corpus), as a result, GSDMM can converge really fast. We can also see the standard deviations of the 20 runs of GSDMM are almost zero which means the performance of GSDMM is highly stable.

4.5 Influence of K

In this part, we try to investigate the influence of K to the number of clusters found by GSDMM and the performance of GSDMM. We fix the number of iterations at 10, $\alpha = 0.1$, and $\beta = 0.1$ for all datasets.

Figure 5 shows the number of clusters found by GSDMM with different values of K on the four datasets. From Figure 5, we can see the number of clusters found by GSDMM grows when we enlarge K, and can finally get stable. The stable number of clusters found by GSDMM is near the ground true number of groups which means GSDMM can infer the number of clusters automatically when K is large enough.

Figure 6 shows the performance of GSDMM with different values of K on the TitleSnippetSet, and we have similar results on the other three datasets. Intuitively, we know a clustering algorithm will tend to have high completeness and low homogeneity when K is small. This is because when K is small, the algorithm can more easily assign the documents belonging to the same ground true group to the same cluster (high completeness), while more difficult to divide the documents in different ground true groups into different clusters (low homogeneity). Similarly, a clustering algorithm will tend to have high homogeneity and low completeness when K is large. From Figure 5, we can see the performance of GSDMM conforms to the above intuition when K is small. Different from the above intuition, GSDMM can achieve both high completeness and high homogeneity when K is

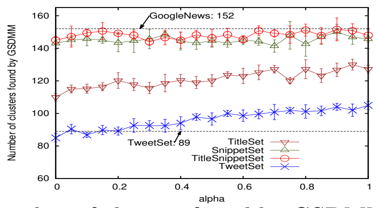

Figure 7: Number of clusters found by GSDMM with different values of α on the four datasets.

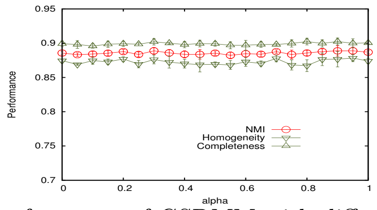

Figure 8: Performance of GSDMM with different values of α on the TweetSet.

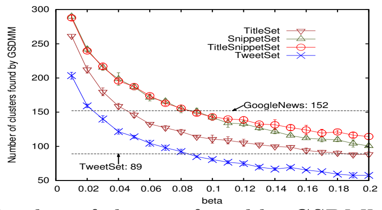

Figure 9: Number of clusters found by GSDMM with different values of β on the four datasets.

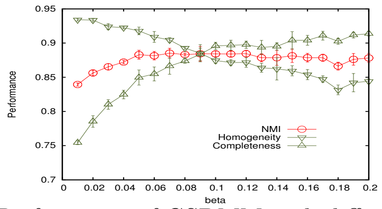

Figure 10: Performance of GSDMM with different values of β on the TweetSet.

larger than the true number of clusters. This is because GSDMM can balance the completeness and homogeneity of the clustering results with Equation 4 which relates to the two rules of MGP as discussed in Section 2.3.

4.6 Influence of Alpha

In this part, we try to investigate the influence of α to the number of clusters found by GSDMM and the performance of GSDMM. We fix the number of iterations at 10, the initial number of clusters at 300, and $\beta = 0.1$ for all datasets.

Figure 7 shows the number of clusters found by GSDMM with different values of α on the four datasets. We can see the number of non-empty clusters found by GSDMM grows slightly with the increase of α on the TweetSet and TitleSet. We investigate the clustering results of these two datasets, and find that larger α tends to result in clusters with only one document. We can explain this phenomenon with the Movie Group Process (MGP). In the extreme case, when we set $\alpha = 0$, a table will be discarded after it gets empty. Because the first part of Equation 4 is now zero for empty tables, and a student (document) will never choose an empty table (cluster). When α gets larger, a student will have a larger probability to choose an empty table. However, once a table gets empty, it will have a low probability of growing large because the “richer gets richer” property of MGP. As a result, the number of non-empty clusters found by GSDMM will get larger slightly with the increase of α, and GSDMM will result in more clusters with only one document. We can see that the number of non-empty clusters found by GSDMM tends to be stable in the SnippetSet and TitleSnippetSet. This is because the average length of the documents is larger in these two datasets, and the second part of Equation 4 plays a more important rule. The students are more likely to choose a table according to their interests, as a result, the empty tables have a lower probability of being chosen.

Figure 8 shows the performance of GSDMM with different values of α on the TweetSet, and we have similar results on the other three datasets. From Figure 8, we can see the performance of GSDMM is stable with different values of α. We also conducted experiments by ranging α from 1 to 10, and found that the performance of GSDMM drops slowly when α is larger than one. We conducted experiments to investigate the performance of GSDMM when $\alpha = 0$, and found that GSDMM almost does not result in clusters with only one document when $\alpha = 0$. We should note that we can also speed up GSDMM by setting α to zero. Because when $\alpha = 0$, we can just discard clusters (tables) that get empty as they will never be chosen again, which means K can decrease with the running of GSDMM. In Section 3.4, we analyzed the time complexity of GSDMM, which is $O(KD\bar{L})$. We can see that GSDMM can get faster with the decrease of K. We should note that Equation 4 cannot be derived from DMM when $\alpha = 0$, because α is the parameter of the Dirichlet distribution in DMM and must be larger than zero. This means the new algorithm should be related to another model, and we plan to investigate this more deeply in future.

4.7 Influence of Beta

In this part, we try to investigate the influence of β to the number of clusters found by GSDMM and the performance of GSDMM. We fix the number of iterations at 10, the initial number of clusters at 300, and $\alpha = 0.1$ for all datasets.

Figure 9 shows the number of clusters found by GSDMM with different values of β on the four datasets. From Figure 9, we can see the number of clusters found by GSDMM drops when we enlarge β. This is because β is in the second part of Equation 4 which relates to Rule 2 of MGP (Choose a table whose students share similar interests with him). When β is small, the probability of a student choosing a table is more sensitive to $n^{w}_{z,\neg d}$ (the number of occurrences of word w in cluster z). In other words, GSDMM gives more emphasis on the student's interests when β is small, and the students will have a larger probability to choose a table based on their interests rather than the popularity of tables. As a result, GSDMM will result in more non-empty tables (clusters) when β is small. Similarly, when β is large, GSDMM will result in fewer non-empty tables.

Figure 10 shows the performance of GSDMM with different values of β on the TweetSet, and we have similar results on the other three datasets. From Figure 10, we can see the completeness of GSDMM grows when we enlarge β. We can understand this phenomenon through Figure 9, where the number of clusters found by GSDMM drops when we enlarge β. As a result, students (documents) in the same ground true group are more likely to be in the same table (cluster), and GSDMM will result in larger completeness when we enlarge β. Similarly, we can understand why the homogeneity of GSDMM drops when we enlarge β. Therefore, β can be used for adjusting the performance of GSDMM. GSDMM can result in relatively more clusters and high homogeneity with small β, and relatively fewer clusters and high completeness with large β.

4.8 Scalability of GSDMM

In this part, we compare the scalability of GSDMM with DMAFP, K-means, and Mini-batch K-means [26]. Mini-batch K-means [26] is a variant of K-means which uses mini-batches to reduce the computation time. Different from K-means, every time Mini-batch K-means assigns a document to a cluster, it updates the centroid of that cluster. We should note that our GSDMM algorithm also updates the model every time it assigns a document to a cluster.

To the best of our knowledge, there is no large short text datasets that can be used to test the scalability of the clustering models. As a compromise, we copied the TitleSet 2 times, 4 times, 8 times,...,until 256 times. By this way, we obtained nine different datasets scaling from 11,109 documents to 2,843,904 documents. For GSDMM, we set $K = 300, \alpha = 0.1, \beta = 0.1$, and run 10 iterations for each dataset. For DMAFP, we set $K = 300$ and the maximum iteration number at 10. For K-means and Mini-batch K-means, we set $K = 152$ (the true number of clusters) and the maximum iteration number at 10. For Mini-batch K-means, we set the mini-batch size at one percent of the number of documents in each dataset. The experiments were conducted on a computer with 8G RAM and 3.4GHz Processor Speed.

Table 3 shows the time cost of GSDMM, DMAFP, K-means, and MiniBatch K-means on these nine datasets. First, we can see the running time of GSDMM, DMAFP, and K-means are both linear to the number of documents. Second, GSDMM and DMAFP can scale to the T256 set which contains 256 copies of the TitleSet with about 284 million titles. However, K-means and MiniBatch K-means cannot scale to the T256 set. Third, We can see that MiniBatch K-means is much faster than GSDMM, DMAFP, and K-means, especially on large datasets. We plan to explore how to speed up GSDMM in future. One promising direction is to utilize the incremental clustering approach discussed in Section 3.2. We can run GSDMM on a subset of the large dataset, then use the Naive Bayes Classifier (NBC) learned by GSDMM to classify other documents into the existing clusters. This approach has potential applications for texts like tweets and news that have many duplicates.

5. RELATED WORK

In this section, we review the related work from the following two perspectives: model-based clustering and short text clustering.

5.1 Model-based Clustering

Nigam et al. [20] used the Dirichlet Multinomial Mixture (DMM) model for classification with both labeled and unlabeled documents, when only unlabeled documents are provided, DMM turns out to be a clustering model. In [20], they proposed an EM-based algorithm for DMM (abbr. to EM-DMM). In this paper, we proposed a collapsed Gibbs Sampling algorithm for DMM (abbr. to GSDMM), and found that GSDMM can infer the number of clusters automatically with a good balance between the completeness and homogeneity of the clustering results, and is fast to converge. Yu et al. [30] proposed the DPMFS model for text clustering that can also infer the number of clusters automatically. However, DPMFS is slow to converge as discussed in [11]. In [11], they further proposed the DMAFP model as an approximation to the DMPFS model and a variational inference algorithm for DMAFP. They compared DMAFP with other four clustering models: EM-DMM [20], K-means [13], LDA [6], and EDCM [7]. They found that DMAFP can achieve better performance than these models especially when the number of clusters given for these models is imprecise. In the experimental study, we compared our GSDMM algorithm with DMAFP, and found that GSDMM can perform better than DMAFP on all the four datasets. In addition, We should note that GSDMM is much simpler than DMAFP, and we can understand how and why GSDMM works as well as the meaning of its parameters with the analogy of MGP.

5.2 Short Text Clustering

Rangrej et al. [21] compared three clustering algorithms for short text clustering: K-means, Singular Value Decomposition and Affinity Propagation on a small set of tweets, and found that Affinity Propagation [8] can achieve better performance than the other two algorithms. However, the complexity of Affinity Propagation is quadratic in the number of documents, which means Affinity Propagation cannot scale to huge datasets with millions of documents. Banerjee et al. [2] proposed a method of improving the accuracy of short text clustering by enriching their representation with additional features from Wikipedia. This method can be a complement for our GSDMM algorithm. Shou et al. [27], Tsur et al. [28], and Yin [29] focused on the problem of online clustering of a stream of tweets. They all utilized an incremental clustering framework that first groups a number of tweets into clusters, then assigns the newly arriving tweets to these clusters. In this paper, we primarily focus on clustering a static collection of short documents, however, we discussed how to use GSDMM for incremental clustering in Section 3.2. We plan to explore this in future.

6. CONCLUSION

In this paper, we proposed a collapsed Gibbs Sampling algorithm for the Dirichlet Multinomial Mixture model for short text clustering (abbr. to GSDMM). We also proposed the Movie Group Process (MGP) as an analogy of GSDMM which can help us understand how and why GSDMM works as well as the meaning of its parameters. We found that GSDMM can infer the number of clusters automatically with a good balance between the completeness and homogeneity of the clustering results, and is fast to converge. GSDMM can also cope with the sparse and high-dimensional problem of short texts, and obtain the representative words of each cluster. Thorough experimental study shows GSDMM can achieve significantly better performance than the baseline methods.

There are several future directions for us to explore. First, we plan to investigate how to incorporate the global weighting metrics (e.g., IDF) for words through β as discussed in Section 3.1. Second, as discussed in Section 4.6, GSDMM can achieve good performance and lower time complexity when we set $\alpha = 0$. However, now GSDMM cannot be de-

	Datasets	T1	T2	T4	T8	T16	T32	T64	T128	T256
	#Docs	11,109	22,218	44,436	88,872	177,744	355,488	710,976	1,421,952	2,843,904
Time	GSDMM	0.092	0.179	0.354	0.711	1.424	2.836	5.688	11.346	23.203
	DMAFP	1.346	2.565	5.090	9.947	19.447	38.991	77.503	154.275	308.651
	K-means	0.399	0.773	1.423	2.954	6.147	11.509	24.072	49.259	N/A
	MiniBatch	0.027	0.032	0.041	0.059	0.094	0.182	0.340	1.197	N/A

Table 3: Running time of the models on nine datasets (in minutes).

rived from DMM because α is the parameter of the Dirichlet distribution in DMM and must be larger than zero. We plan to explore which model this new algorithm is related to in future. Third, we also plan to study how to use GSDMM for incremental clustering as discussed in Section 3.2.

7. ACKNOWLEDGMENTS

This work was supported in part by National Natural Science Foundation of China under Grant No. 61272088, National Basic Research Program of China (973 Program) under Grant No. 2011CB302206, and Tsinghua University Initiative Scientific Research Program.

8. REFERENCES

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

[2] S. Banerjee, K. Ramanathan, and A. Gupta. Clustering short texts using wikipedia. In *SIGIR*, pages 787–788, 2007.

[3] H. Becker. *Identification and Characterization of Events in Social Media*. PhD thesis, COLUMBIA UNIVERSITY, 2011.

[4] P. Berkhin. A survey of clustering data mining techniques. In *Grouping Multidimensional Data*. 2006.

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

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

[7] C. Elkan. Clustering documents with an exponential-family approximation of the dirichlet compound multinomial distribution. In *ICML*, 2006.

[8] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *science*, 315(5814):972–976, 2007.

[9] G. Heinrich. Parameter estimation for text analysis. Technical report version 2.9, vsonix GmbH and University of Leipzig, 2009.

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

[11] R. Huang, G. Yu, Z. Wang, J. Zhang, and L. Shi. Dirichlet process mixture model for document clustering with feature partition. *IEEE Trans. Knowl. Data Eng.*, 25(8):1748–1759, 2013.

[12] L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

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

[14] R. E. Madsen, D. Kauchak, and C. Elkan. Modeling word burstiness using the dirichlet distribution. In *ICML*, pages 545–552, 2005.

[15] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge, 2008.

[16] A. McCallum, K. Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer, 1998.

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

[18] A. Y. Ng, M. I. Jordan, Y. Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2, 2002.

[19] X. V. Nguyen, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11:2837–2854, 2010.

[20] 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.

[21] A. Rangrej, S. Kulkarni, and A. V. Tendulkar. Comparative study of clustering techniques for short text documents. In *WWW (Companion Volume)*, pages 111–112, 2011.

[22] J. D. Rennie. *Improving multi-class text classification with naive Bayes*. PhD thesis, Massachusetts Institute of Technology, 2001.

[23] J. D. Rennie, L. Shih, J. Teevan, and D. R. Karger. Tackling the poor assumptions of naive bayes text classifiers. In *ICML*, pages 616–623, 2003.

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

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

[26] D. Sculley. Web-scale k-means clustering. In *WWW*, pages 1177–1178, 2010.

[27] L. Shou, Z. Wang, K. Chen, and G. Chen. Sumblr: continuous summarization of evolving tweet streams. In *SIGIR*, pages 533–542, 2013.

[28] O. Tsur, A. Littman, and A. Rappoport. Efficient clustering of short messages into general domains. In *ICWSM*, 2013.

[29] J. Yin. Clustering microtext streams for event identification. In *IJCNLP*, pages 719–725, 2013.

[30] G. Yu, R. Huang, and Z. Wang. Document clustering via dirichlet process mixture model with feature selection. In *KDD*, pages 763–772, 2010.