Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Text Clustering Approach Based on Maximal Frequent Term Sets

Chong Su, Qingcai Chen, Xiaolong Wang, Xianjun Meng
Intelligence Computing Research Center, Shenzhen Graduate School
Harbin Institute of Technology
Shenzhen, China
chong.su@hotmail.com, qingcai.chen@gmail.com, wangxl@insun.hit.edu.cn, mengxj@insun.hit.edu.cn

***Abstract*—Classical text clustering algorithms are usually based on vector space model or its variants. Because of the high computing complexity and the difficulty of controlling clustering results, this kind of approaches are hard to be applied for the purpose of the large scale text clustering, Clustering algorithms based on frequent term sets make use of relationship among documents and their shared frequent term sets to achieve high accuracy and effectiveness in clustering. But since the number of frequent terms is usually too large to reach the efficiency requirement for large collection texts clustering, this paper proposes a novel text clustering approach based on maximal frequent term sets (MFTSC). This approach firstly mines maximal frequent term sets from text set and then clusters texts by following steps: at first, the maximal frequent term sets are clustered based on the criterion of k-mismatch; then texts are clustered according to term sets clustering results; finally, we categorize the left texts uncovered in previous step into produced text clusters Be compared with existing approaches, our experimental results show an average gain of 10% on F-Measure score with better performance on scalability and efficiency.**

***Keywords*—Text Clustering, Text mining, Frequent Term Sets, Maximal Frequent Term Sets**

I. INTRODUCTION

With the rapid development of the Internet, there are a huge number of texts available on Word Wide Web. How to efficiently organize and make use of these resources has been a hot research topic in field of data mining and knowledge discovering. Text clustering is an important method of organizing information and has the ability of helping users to understand the intrinsic structure of information by unsupervised way. Existing text clustering algorithms usually represent texts by some variants of basic Vector Space Model (VSM). And most of the text clustering algorithms, such as hierarchical agglomerative [1], probabilistic clustering [2], k-means [1], bisecting k-means [1], self-organizing maps [3] etc., are derived from structured data clustering algorithms. Since the high dimensionality of text vectors and uncontrollable of clustering results based on vector space model, they are not so suitable for the automatic organization of large scale text collections.

Compared with vector space models, the document clustering models based on frequent term sets (FTS)[4] that had been proposed in recent years are more easier to control the clustering results and also more efficient. Here the frequent term sets are defined as the sets of terms co-occurring in more than a threshold percentage of all documents [4]. For example, let D be the text documents set and {movie, actor, actress, star} occurs more than 5% of all the documents in D, so we call this term set is frequent term set according to the threshold of 5%. The set of documents which sharing the term set is called supporting documents. The basic idea in FTS based text clustering is that the documents that contain the same FTS tend to be relevant and then should be grouped together. Since the clustering algorithm is just relying on the relationship of sharing on FTS, the dimensionality of algorithm features and the computational complexity of clustering are greatly reduced. And the shared FTS also provide good candidates to label text clusters for the human understanding purpose.

To efficiently implement the basic idea of FTS based text clustering, various algorithms had been proposed. The FTC (Frequent Term-based Clustering) [4] algorithm mines all frequent term sets and generates cluster candidates according to the frequent term sets' supporting documents sets. To discover a clustering with a minimum overlap of the clusters, it follows a greedy approach, i.e., it repeatedly picks up the candidate cluster of which the supporting document sets has minimum overlap with other candidates and the procedure is terminated after all documents are covered. Though some extent of successful, FTC just considers the overlap of FTS' supporting document sets, and the greedy approach can not ensure the global optimum. The Frequent Itemset-based Hierarchical Document Clustering (FIHC) [5] algorithm firstly generates cluster candidates from all frequent term sets. Some documents may belong to more than one cluster. It then judges "best" cluster according to a score function, merges clusters and generates hierarchical representations. However, FIHC has high time and space complexity, especial for small threshold of its min support value. On the other hand, small threshold for min support value is necessary for the clustering algorithms to achieve higher precision. So the balancing between precision and scalability is a critical issue for FIHC algorithm.

In this paper, we proposed a novel text clustering approach based on maximal frequent term sets (MFTSC). The unique of our approach is listed below:

At first, the proposed approach only relies on maximal frequent term sets (MFTS). Instead of using all FTS, just using

MFTS can greatly reduce the time consuming of text clustering. The MFTS also tend to be longer and be more meaningful for similarity measurement in text clustering.

Then a two-stage clustering process is executed in our approach. Even though we only use MFTS, it may still be too complex for large scale of text collections to execute the clustering task. On the other hand, generating cluster candidates from all MFTS will cause a great amount of redundancies. To address this problem, we cluster the MFTS firstly. Then we define the supporting document set of a MFTS cluster as the union of supporting document sets of all the MFTS in this cluster. Documents are then being clustered based on these supporting document sets.

Finally, the uncovered documents are clustered by categorizing. After the two-stage clustering, some texts may not be covered by any MFTS. These texts are finally classified into existing text clusters.

The rest of this paper is organized as follows: Section 2 briefly introduces the concepts of FTS and the related FTS mining algorithms. In section 3, the text clustering approach based on maximal frequent term set is presented in detail. The experimental results are shown in section 4 accompanied with analysis. Finally we summarize our work and future research issues in section 5.

II. Frequent Term Sets

Frequent term set and its mining algorithm are the foundation of our approach. Here we just give a brief introduction for their basic conceptions. For detail information about FTS, please refer to [6].

A. Definition of Frequent Term Sets

Let $I=\{I_1, I_2 \ldots I_n\}$ be a set of n-terms, a set $X \subseteq I$ is called a term set, or a *k*-term set if it contains k terms.

Let $D=\{D_1, D_2 \ldots D_m\}$ be a set of documents, $D_j \subseteq I$. The support of a term set X in D is the proportion of documents that contained X among the document set D, denoted as Supp(X). The user can specify a minimum support value min_supp as a threshold for selecting frequent term sets as below:

For the documents set D and minimum support min_supp, the term set X is called frequent term set (FTS), if Supp(X) > min_supp.

For the documents set D and a given minimum support min_supp, the term set X is called maximal frequent term set (MFTS), if X is frequent term set and for $\forall (Y \subseteq I \wedge X \subseteq Y)$, Supp($Y$) < min_supp.

To illustrate above concepts, let's see an example. Let $I=\{a, b, c, d, e\}$, $D=\{D_1, D_2, D_3, D_4\}$ and four documents as $D_1=\{a, b, c, e\}$, $D_2=\{a, b, c\}$, $D_3=\{c, d\}$, $D_4=\{c, d, e\}$. We set the min support threshold be 0.5. Then {a, b} is a 2-term set and FTS. But {a, b} is not a MFTS because {a, b, c} is {a, b}'s superset and {a, b, c} is frequent too. And the {a, b, c}'s supporting documents set is $\{D_1, D_2\}$.

B. Frequent Term Sets Mining

Apriori is the first frequent term sets mining algorithm proposed by Agrawal et al [7]. It generates candidate k-term sets from (k-1)-term sets and then checks if they are frequent or not. But the time complexity of Apriori is too high to be applied for large scale text collections. So in 2000, Han et al [8] proposed the FP-Growth algorithm that mining frequent term sets without generating candidate.

Based on FP-Growth, FP-Max algorithm in [9] improved the efficiency of the mining process by only considering the maximal frequent term sets. In this paper, the FP-Max algorithm is also applied to mine maximal frequent term sets from texts.

III. The Proposed Clusteing Approach

In this section we present the approach of clustering documents based on MFTS. This approach consists of 3 stages. At first, we cluster the MFTS based on the k-mismatch metric defined on term sets. Then document clusters are produced according to the FTS clusters' support documents and then merge the clusters whose document sets have a higher overlap. Some documents may not be covered by any of the MFTS and then can not be covered by any cluster. So our final stage is classifying these documents into existing document clusters according to the similarity between document and the cluster.

Before presenting the approach, some notations are defined firstly. Here we denote the set of MFTS as $MFS=\{MF_1, MF_2 \ldots MF_m\}$, set of FTS clusters as $FCS=\{FC_1, FC_2 \ldots FC_n\}$, set of document clusters as $DCS=\{DC_1, DC_2 \ldots DC_k\}$.

A. Frequent Term Set Clustering Based on K-mismatch

Usually there are hundreds or more MFTS for a large scale text collection. Generating cluster candidates directly by all MFTS will get a great amount of redundancies. Table I shows a small part of MFTS mined from a document set. Our idea of clustering MFTS is based on the observation that MFTS can be considered as short documents.

TABLE I. Example of Maximal Frequent Term Set

NO.	Maximal frequent term set	Supporting document set
1	{move, actor, actress, director, role}	{1,2,3,4,5,6,7}
2	{move, actor, actress,star}	{1,2,3,4,8,9,10,11}
3	{actor, actress, director, role,star}	{1,2,5,6,7,9,11,12,13}
4	{star, eath, planet,moon, satellite}	{14,15,16,17,18,19,20}
5	{star, eath, planet,moon, human }	{15,17,18,19,20,21,22,23}
6	{human, planet, satellite}	{18,19,20,23,24,25,26}

The MFTS clustering algorithm is based on k-mismatch concept [11]. In our case, the k-mismatch term set of term set A is defined as follow:

$$K - Mismatch(A) = \{S \in MFS \| S - S \cap A | \leq K\} \qquad (1)$$

For example, we have A= {a, b, c, d, e}, B= {a, c, d, e, f}, C= {b, f, g, h} and k is 2, then B is A's k-mismatch term set because B only has one term “f” that doesn’t present in A, but C is not A’s k-mismatch term set.

Longer term sets which contain more terms may indicate more specific topic, so the supporting documents are more likely belonged to one cluster. The MFTS clustering algorithm begins with the longest term set and is described in Algorithm 1.

```
Algorithm 1: Frequent term sets clustering algorithm
Input: MFTS set MFS, min longest threshold h
Output: MFTS cluster set FCS
Let A =longest term set in MFS
While |A| > h do:
    Generate FC_i by A , i++
    Remove A out from MFS
    For each MF_j in MFS do:
        If it is A's k-mismatch term set, then
            Put MF_j into FC_i
            Remove MF_j from MFS
    Let A =longest term set in MFS
```

The algorithm iteratively picks the longest term set A, group A and A’s k-mismatch term sets into a cluster and then remove A and A’s k-mismatch term sets from MFS. When the length of remained longest term set is less than a threshold *h* (usually *h*=2*k*), the procedure is terminated.

TABLE II. EXAMPLE OF MAXIMAL FREQUENT TERM SET CLUSTERS

No.	Max frequent term sets	Supporting document set
1	1,2,3	{1,2,3,4,5,6,7,8,9,10,11,12,13}
2	4,5	{14,15,16,17,18,19,20,21,22,23}
3	6	{18,19,20,23,24,25,26}

Table II shows an example of MFTS clusters derived from Table I. The support set of each MFTS cluster is the union of support of all MFTS.

B. Documents Clustering Based on MFTS Clusters

After the first stage, we get the MFTS clusters, with the corresponding term sets and support document sets. Document clusters *DCS* are generated by the MFTS clusters *FCS*, and each document cluster is composed by the support document set of each MFTS cluster. But there are still too many clusters and many redundancies existing in these clusters.

To remove redundancies and get the final clusters, some clusters need to be merged. The merger of clusters is based on the overlap of two clusters’ supporting document set. There are many variants for set overlap definitions. Here we build our calculation of set overlap based on the basic definition given below:

Let cov(DC_i) denoted the supporting document set covered by cluster DC_i, then we have:

$$overlap(DC_i, DC_j) = \frac{|\operatorname{cov}(DC_i) \cap \operatorname{cov}(DC_j)|}{\min(|\operatorname{cov}(DC_i)|, |\operatorname{cov}(DC_j)|)} \quad (2)$$

Above general definition may cause “Matthew effect” in clustering documents, i.e., documents tend to be merged into a bigger size clusters. To overcome this problem, we develop a new definition for the overlap by introducing a parameter a as fellows:

$$overlap(DCS_i, DCS_j) =$$

$$\frac{|\operatorname{cov}(DC_i) \cap \operatorname{cov}(DC_j)|}{a \times \min(|\operatorname{cov}(DC_i)|, |\operatorname{cov}(DC_j)|) + (1-a) \times \sqrt{|\operatorname{cov}(DC_i)| \times |\operatorname{cov}(DC_j)|}} \quad (3)$$

Where a is the coefficient to balance the “Matthew effect”. In this paper, a is set as 0.7 according to the algorithm performance.

The text clustering procedure iteratively merges those clusters among them the overlap value is greater than the threshold and is terminated until there is no clusters satisfy the merge condition or the number of clusters achieves specified value.

C. Uncovered Docuemnts Classification

In the FTS mining procedure, there is a small part of documents that do not contain any FTS. And even more documents may be missed by MFTS. Our experiment results show that there are about 5%-20% documents are not covered by any cluster and the proportion is depended on the threshold of min support.

Because we have already constructed clusters that cover most of documents, so classifying those uncovered documents into existing document clusters is a natural way to address above problem. For this purpose, we define the similarity between document and cluster as follows:

Let D_i be uncovered document, DC_j is j^{th} cluster and I is the set of frequent terms in DC_j, $tf(D_i, I_k)$ denotes the frequent of term I_k in D_i, then we have:

$$sim(D_i, DC_j) = \frac{1}{\sqrt{|I|}} \sum_{I_k \in I} tf(D_i, I_k) \quad (4)$$

A document is directly classified into the cluster that owns the maximum similarity value with the document.

IV. EXPERIMENTS AND RESULTS

In this section, we evaluate the performance of proposed clustering algorithm in terms of accuracy and scalability. The program is run on Windows XP with Intel Core 2 Quad 2.39GHz CPU, and 2GB memory.

A. *Data Set*

To be comparable, two widely used data sets are applied. One is from the Classic4 data set [12], which contains 5476 documents from 4 classes: CACM, CISI, CRAN and MED. The other data set is from Reuters Distribution 1.0[13], which contains 21578 articles from Reuter news service. The dataset C1, C2, C3, C4 are chosen from Classic4, while R1, R2, R3, R4, R5, R6, R7, R8 are chosen from Reuters. The data sets are different in document size, cluster size, number of classes, and category distributions.

TABLE III. EXPERIMENT DATA SET

Data set	Num of doc.	Num of class.	Min class size	Max class size	Avg class size	Avg doc. length	Num of unique term
R1	450	5	39	142	90	114	4424
R2	2371	6	36	1954	455	78	8738
R3	268	6	20	106	44	124	3427
R4	206	6	18	61	34	84	2390
R5	802	10	8	301	80	136	5570
R6	4061	10	33	2494	406	75	6169
R7	1644	13	16	581	126	105	7121
R8	3750	24	13	1616	156	84	12018
C1	407	4	87	118	101	67	4885
C2	1273	4	247	365	318	66	8080
C3	428	4	41	229	107	76	5544
C4	597	4	47	432	149	65	5602

B. *Evaluation Measure*

In this paper, the widely used F is adopted for the algorithm evaluation. Each cluster is as if it was the result of a query and each class is treated as if it was the relevant documents set of a query [1].

For cluster j and class i, the recall and precision are defined below:

$$Recall(i,j) = \frac{n_{ij}}{n_i} \tag{5}$$

$$Precision(i,j) = \frac{n_{ij}}{n_j} \tag{6}$$

Where n_{ij} is the number of documents in class i which are clustered in cluster j, n_i is the number of documents in class i and n_j is the number of documents in cluster j.

The F measure of cluster j and class i is defined as follow:

$$F(i,j) = \frac{2 * Recall(i,j) * Precision(i,j)}{Recall(i,j) + Precision(i,j)} \tag{7}$$

For the entire clustering result, the F measure is obtained by using the weighted sum of maximum F measure in each class.

$$F = \sum_i \frac{n_i}{n} \max\{F(i,j)\} \tag{8}$$

Where n is the total number of documents.

C. *Experimental Results*

1) Accuracy: To compare our approach's accuracy, we also executed bisecting k-means (BKM) [1], and FIHC [5] on the same data sets. We chose BKM because it outperforms k-means and agglomerative hierarchical clustering algorithms, according to paper [1]. FIHC is chosen because it is based on frequent term set too. We obtained FIHC version 1.0 from the inventor of FIHC and downloaded the CLUTO toolkit [14] to perform BKM.

The same number of clusters for all the algorithms is given. Because the performance of BKM is in some extent depended on the initial setup, so we execute 10 times and then take the average on F-measures. For each run of BKM, it randomly selects initial point and picks biggest cluster to divide. The FIHC is executed with the default parameters provided in [5]. The min support of FIHC is set to 0.05.

One advantage of the proposed MFTSC approach is that under the same limitation of computing complexity, it can cluster documents with smaller min support than FIHC and then is expected to get more precision results. Here we set the min support for each group of data as the smallest min support that every data set in the group can be executed normally, i.e., the min support for Classic group is 0.01 and 0.03 for Reuters.

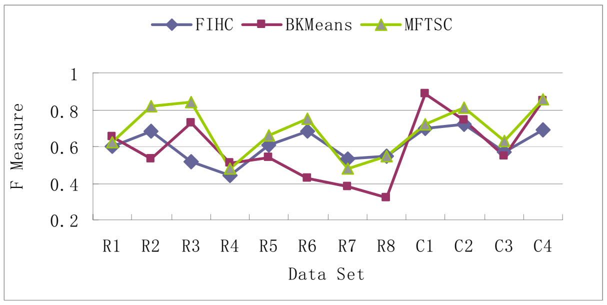

Figure 1. Comparison of clustering algorithms' accuracy

Figure 1 shows that the MFTSC outperforms BKM and FIHC in most data sets and has great improvement than BKM for large number of classes (for example, R5, R6, R7, R8).

The BKM performs well when the number of classes is small, but has a worse performance than both MFTSC and FIHC when number of clusters is increased. Compared with FIHC, MFTSC can mine frequent term sets with a smaller support threshold, so MFTSC can get frequent term sets combination of more terms and the average length of frequent term sets is longer. With more terms and longer frequent term sets, it may reveal more and stronger relationship between documents. It is helpful for similarity calculation among documents. The experimental results show that MFTSC has an average 10% improvement over FIHC.

The min support is an important factor in FTS based clustering. For a given data set, smaller min support means more frequent term sets been mined and better accuracy performance is expected respectively. At the same time, it also takes more clustering time for smaller min support (as shown in Figure 2). We conduct our experiment on the Reuter data set with the min support value various from 0.03 to 0.05 and show the results in Table IV. The results show that MFTSC performs better with the smaller min support in most case.

TABLE IV. EFFECT OF MIN SUPPORT ON F MEASURE OF MFTSC

Data Set	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**Avg**
0.03	0.62	0.82	0.84	0.48	0.66	0.75	0.48	0.55	**0.65**
0.04	0.60	0.69	0.85	0.55	0.67	0.70	0.53	0.44	**0.63**
0.05	0.46	0.64	0.85	0.68	0.67	0.74	0.51	0.38	**0.62**

2) *Efficiency and Scalability:* The running time of FIHC and MFTSC are compared with various min supports and results are shown by Figure 2. Since MFTSC only takes use of MFTS, it is expected to comsume less time than FIHC for the same min support value. The experiment is executed on the data set R6 that contains 4061 documents.

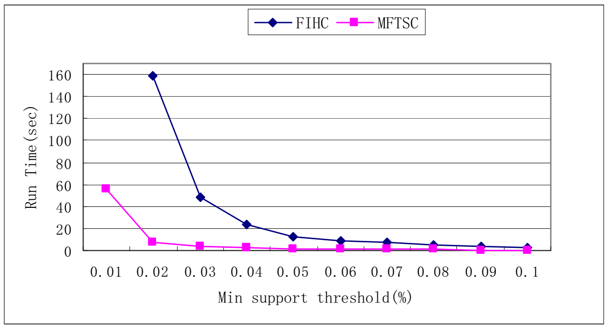

Figure 2. Running time of FIHC and MFTSC with respect to documents size

The results shown on Fig. 2 proved our expectation. It should be noted that with the increasing of min support value, the difference of time consuming for these two algorithms become not so significant. But for smaller min support, the advantage of MFTSC becomes more obvious. E.g., for the min support of 0.02, the running time of MFTSC is 7.5 seconds while which of FIHC is 159 seconds. The running time of FIHC with the 0.01 min support can't be obtained because the program runs out of memory and terminated with exception.

To evaluate the scalability of MFTSC, we randomly select documents from Reuter dataset to create 10 data sets with the number of document increase from 1000 to 10000. Two min support values 0.03 and 0.05 are experimented. Fig. 3 shows that the running time of both FIHC and MFTSC is linear with respect to the number of documents. But the running time of MFTSC grows at a lower rate and the time advantage is more obvious when the number of documents is larger.

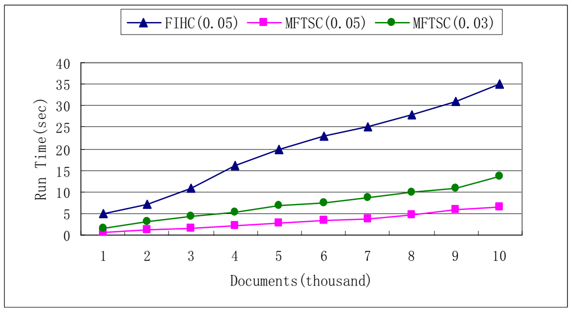

Figure 3. Running time of FIHC and MFSTC with respect to document size

V. CONCLUSION

This paper presents a text clustering approach based on frequent term sets. Compared with existing clustering algorithms based on frequent term sets, the proposed algorithm MFTS is only based on the maximal frequent term sets and two-stage clustering procedure is proposed for the frequent term sets and the documents clustering correspondingly. The uncovered documents are finally classified into existing clusters based on document-cluster similarity. Our experiment results show the improvement of our approach on both efficiency and scalability.

There are still a lot of issues for future research, which include but not limited to the refinement of overlap measure for document clustering, the document-cluster similarity computing and the efficiency improvement of MFTSC for huge scale of document collections.

ACKNOWLEDGMENT

This investigation was supported in part by the National Natural Science Foundation of China (No. 60703015) and the National 863 Program of China (No. 2006AA01Z197).

REFERENCES

[1] M. Steinbach, G. Karypis, V. Kumar. "A comparison of document clustering techniques". Technical Report, University of Minnesota ,2000: 00-34.

[2] S.F. Zhu, I. Takigawa, S.Q. Zhang, H. Mamitsuka. "A probabilistic model for clustering text documents with multiple fields". 29th European Conference on IR Research, Proceedings. 2007: 331-342.

[3] R. Freeman, H.J. Yin. "Self-organizing maps for hierarchical tree view document vlustering using contextual information". Proceedings of the IEEE International Joint Conference on Neural Networks. 2002:123-128.

[4] F. Beil, M. Ester, X.W. Xu. "Frequent term-based text clustering". Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2002:436-442.

[5] B.C.M. Fung, K. Wang, M. Ester. "Hierarchical document clustering using frequent itemsets". Proceedings of SIAM International Conference on Data Mining". 2003:59-69.

[6] Bart Goethals. "Survey on frequent pattern mining". Technical Report: Helsinki Institute for Infrmation Technology, 2003:1-5.

[7] R. Agrawal, R. Srikant R. “Fast algorithms for mining association rules in large databases”. Proc. VLDB 94, Chile, 1994: 487-499.

[8] J. Han, J. Pei, Y.W. Yin, R. Mao. “Mining frequent patterns without candidate generation”. Proceeding of Special Interest Group on Management of Data. 2000:1-12.

[9] G. Grahne, J.F. Zhu. “High performance mining of maximal frequent itemsets”. Proceedings of the 6th SIAM International Workshop on High Performance Data Mining. 2003:311-337.

[10] E.H. Han, D. Boley, M. Gini, R. Gross, et al. “WebACE: A web agent for document categorization and exploration”. Conference on Autonomous Agents. 1997:408 – 415.

[11] G.M. Landau, U. Vishkin, “Fast parallel and serial approximate string matching”, Journal of Algorithms, 10 (2). 1989:157–169.

[12] ftp://ftp.cs.cornell.edu/pub/smart.

[13] http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html.

[14] G. Karypis. “Cluto 2.0 clustering toolkit”, April 2002. http://www-users.cs.umn.edu/˜ karypis/cluto/.