Document Clustering Method Based on Frequent Co-occurring Words

Ye-Hang Zhu[1], Guan-Zhong Dai[1], Benjamin C. M. Fung[2], De-Jun Mu[1],

[1] College of Automation, Northwestern Polytechnical University, Xi'an 710072, China
[2] School of Computing Science, Simon Fraser University, BC, Canada, V5A 1S6
zhuyehang@yahoo.com.cn

Abstract. This paper presents a new document clustering method based on frequent co-occurring words. We first employ the Singular Value Decomposition, and then group the words into clusters called word representatives as substitution of the corresponding words in the original documents. Next, we extract the frequent word representative sets by Apriori. Subsequently, each document is designated to a basic unit described by the frequent word representative set, from which we can get the ultimate clusters by hierarchical clustering. The major advantage of our method is that it can produce the cluster description by the frequent word representatives and then by the corresponding words in the clustering process without any extra works. Compared with the state-of-the-art UPGMA method on benchmark datasets, our method has better performance in terms of the entropy and cluster purity.

Keywords: document clustering, text clustering, frequent itemsets, Apriori.

1 Introduction

In recent years, the volume of text has growth tremendously due to the Internet, digital libraries, news sources, and company-wide intranets. There is an urgent need to organize them in a structure that facilitates browsing and searching. Clustering [5] is a classic area of machine learning and pattern recognition. There are three major challenges for clustering (hyper)text databases [2]: High dimensionality of the data, large database, and lack of intuitive cluster description. In this paper, we present a novel clustering algorithm based on the idea of frequent words to address these challenges. Fung et al. [4] and Beil et al. [2] introduced a new criterion for hierarchical document clustering using frequent itemsets.

In this paper, we use frequent word sets together to illustrate one ultimate cluster. For example, in first cluster the word 'data' appear alone and 'mine' appear alone also, cannot form frequent 2 word sets, in second cluster the word 'data' and 'mine' often appear, form frequent 2 word sets, the meaning of the first cluster is much more harder to guess than the second cluster, because when the two word 'data' and 'mine' constitute phrase 'data mine', the meaning of phrase 'data mine' is very concrete and definite, so people is easy to associate the second cluster with the concrete meaning: data mine, but the meaning that word 'data' alone and word 'mine' alone in the first cluster can give is not so definite, so frequent 2 word sets can provide much more information than the word alone that constitute it.

2 Our Clustering Algorithm

Below, we describe our clustering algorithm with an example.

Step 1 (Preprocessing): Our method employs several preprocessing steps including stop words removal and stemming on the document set. We use SVD(Singular Value Decomposition) on the matrix of word and documents to group similar words to the same cluster [8]. After removing high frequency words and low frequency words, we apply TF*IDF [9].

Step 2 (Dimension reduction): If we mine frequent itemset directly from words, the result is not good, because the number of word is large, and the relation between words leads to the number of frequent itemsets is large too, so we cluster the words to word representatives, whose number is small, for example 1000 in our experiment. We replace the words in original files by these word representatives.
Step 3 (Data size reduction): To reduce the size of the data, every file in original data set will retain at most 20 word representatives which are more frequent than the other word representatives in this file. After the transformation, the file size is drastically reduced.
Step 4 (Sampling frequent words representative sets): Then we use Apriori [1] to mine the frequent itemsets from the reduced dataset, and sample at most 500 frequent word representatives from each level of frequent itemsets, and used these word representatives as identifier of every basic unit.
Step 5 (Basic units construction): Our method constructs basic unit in two steps:

Constructing Initial Basic Unit: For each sampled global frequent word representative set, we construct an initial basic unit, the files which include every item in this frequent word representative sets is attached to this basic unit. Table 1 shows the basic unit (cluster). One document may contain several global frequent word representative sets, so one document can be attached to many basic units.

Table 1 Initial basic unit(cluster) with cluster minimum support value 70%

Basic unit (cluster)	Files in basic unit (cluster)	Cluster frequent item and its support value in this basic unit (cluster)
C(flow)	cran.1,cran.2,cran.3,cran.4,cran.5	{flow, CS=100%} {layer,CS=100%}
C(form)	cisi.1,cran.1,cran.3,med.2,med.5	{form,CS=100%}
C(layer)	cran.1,cran.2,cran.3,cran.4,cran.5	{layer,CS=100%} {flow,CS=100%}
C(patient)	med.1,med.2,med.3,med.4,med.5,med.6	{patient,CS=100%} {treatment,CS=83%}
C(result)	cran.3,med.1,med.2,med.4,med.6	{result,CS=100%} {patient,CS=80%} {treatment,CS=80%}
C(treatment)	med.1,med.2,med.3,med.4,med.6	{treatment,CS=100%} {patient,CS=100%} {result,CS=80%}
C(flow,layer)	cran.1,cran.2,cran.3,cran.4,cran.5	{flow,CS=100%} {layer,CS=100%}
C(patient, treatment)	med.1,med.2,med.3,med.4,med.6	{patient,CS=100%} {treatment,CS=100%} {result,CS=80%}

Making Basic Unit Disjoint: For each document, we identify the "best" initial basic unit and keep the document only in the best initial basic unit. Suppose that Score(Ci ← doc_j) measures the goodness of a basic unit Ci for a document doc_j . For each doc_j , we remove doc_j from all the initial basic unit Ci that contain doc_j but one for which Score(Ci ← doc_j) is maximized. If there are more than one Ci that maximizes Ci ← doc_j, choose the one that has the most number of items in the basic unit label.

We borrow the score function Score(Ci ← doc_j) from [4]. Intuitively, a basic unit Ci is "good" for a document doc_j if there are many global frequent items in doc_j that appear in "many" documents in Ci. The "many" is qualified by being basic unit frequent in Ci. The following score measures the goodness of an initial basic unit Ci for a document doc_j .

$$Score(C_i \leftarrow doc_j) = \left[\sum_x n(x) * cluster\_frequency(x)\right] - \left[\sum_{x'} n(x') * global\_frequency(x')\right] \quad (1)$$

where x represents a global frequent item in doc_j and the item is also basic unit frequent in Ci; x' represents a global frequent item in doc_j that is not basic unit frequent in Ci; n(x) and n(x') are the weighted frequency of x and x' in the feature vector of doc_j . n(x) and n(x') are defined by the TF×IDF of item x and x', cluster_frequency(x) denotes the ratio between the number of file which include this word x among this basic unit(cluster) and the total number of file attached to this basic unit(cluster), global_frequency(x') denotes the ratio between the number of files which include this word x' among all the original data set and the number of all the files in original data set. The first term of the score function rewards basic unit Ci if a global frequent item x in doc_j is basic unit frequent in Ci. In order to capture the importance (weight) of item x in different basic unit, we multiply the frequency of x in doc_j by its basic unit support in Ci. The second term of the function penalizes basic unit Ci if a global frequent item x' in doc_j is not basic unit frequent in Ci. The frequency of x' is multiplied by its global support which can be viewed as the importance of x' in the entire document set. This part encapsulates the concept of dissimilarity into the score. The ultimate result is shown in Table 2.

Table 2 Ultimate basic units (cluster)

Basic unit (Cluster)	Files in basic unit (cluster)	Cluster frequent item and its support value in this basic unit(cluster)
C(flow)	cran.1,cran.2,cran.3,cran.4,cran.5	{flow, CS=100%} {layer,CS=100%}
C(form)	cisi.1	{form,CS=100%}
C(layer)		
C(patient)	med.5	{patient,CS=100%} {treatment,CS=83%}
C(result)		
C(treatment)		{treatment,CS=100%} {patient,CS=100%} {result,CS=80%}
C(flow,layer)		
C(patient, treatment)	med.1,med.2,med.3, med.4,med.6	{patient,CS=100%} {treatment,CS=100%} {result,CS=80%}

Step 6 (Hierarchical clustering): Calculate average value of the document vector of the files attached to the same basic unit(cluster), use these average value as the value of the corresponding basic unit(cluster), then by hierarchical clustering method we cluster the basic units(clusters) to form the ultimate clusters, at the same time we get the corresponding words in these basic units(clusters)'s label, we can use these words as illustration of the corresponding ultimate cluster, so as to let the people to realize the meaning and character of the corresponding ultimate cluster.

3 Experimental Results

Experiments in Steinbach et al. [10] suggest that UPGMA [5] is one of the most accurate agglomerative hierarchical clustering methods, so we experimentally compare our method with UPGMA in the Cluto software package [6] on a number of benchmark datasets: tr12, tr23, tr45 are derived from TREC-5 [11], TREC-6 [11], and TREC-7 [11] collections. Data sets re0 are from Reuters-21578 text categorization test collection Distribution [7]. Data set wap is from the WebACE project (WAP) [3].

The first metric is the widely used entropy measure [2] that looks how the various classes of documents are distributed within each cluster, and the second measure is the purity that measures the extend to which each cluster contained documents from primarily one class. In general, the smaller the entropy values, the better the clustering solution is. In general, the larger the values of purity, the better the clustering solution is. Table 3 illustrates the results.

Table 3 Comparing with UPGMA

Dataset	Entropy of ours	Entropy of UPGMA	Purity of Ours	Purity of UPGMA
Wap	0.448538	0.449249	0.553846	0.540385
Re0	0.501598	0.606318	0.516622	0.475399
Tr12	0.554687	0.599146	0.549521	0.498403
Tr23	0.428291	0.453157	0.676471	0.686275
Tr45	0.413752	0.514953	0.633333	0.534783

Acknowledgments. We thank the support of CENTER FOR HIGH PERFORMANCE COMPUTING of Northwestern Polytechnical University.

References

1. Agrawal, R., Srikant, R.: Fast algorithm for mining association rules. In: VLDB. (1994) 487-499
2. Beil, F., Ester, M., Xu, X.: Frequent term-based text clustering. In: KDD. (2002)
3. Boley, D., Gini, M. Gross, R., Han, E. H., Hastings, K., Karypis, G., Kumar, V., Mobasher, B., Moore, J.: Document categorization and query generation on the world wide web using WebACE. In: AI Review 11. (1999) 365–391
4. Fung, B. C. M., Wang, K., Ester, M.: Hierarchical Document Clustering Using Frequent Itemsets. In SDM. (2003)
5. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers. (2001)

6. http://glaros.dtc.umn.edu/gkhome/views/cluto
7. http://www.daviddlewis.com/resources/testcollections/
8. Laham, R. D.: Automated Content Assessment of Text Using Latent Semantic Analysis to Simulate Human Cognition, Ph.D. Thesis, University of Colorado, UMI number:9979362
9. van Rijsbergen, C. J.: Information Retrieval. Dept. of Computer Science, University of Glasgow, Butterworth, London, 2nd edition. (1979)
10. Steinbach, M., Karypis, G., Kumar, V.: A comparison of document clustering techniques. In: KDD Workshop on Text Mining'00. (2000)
11. TREC. Text REtrieval conference. http://trec.nist.gov (1999)