International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:4, 2008

A Text Clustering System based on k-means Type Subspace Clustering and Ontology

Liping Jing, Michael K. Ng, Xinhua Yang, and Joshua Zhexue Huang

***Abstract*—** This paper presents a text clustering system developed based on a k-means type subspace clustering algorithm to cluster large, high dimensional and sparse text data. In this algorithm, a new step is added in the k-means clustering process to automatically calculate the weights of keywords in each cluster so that the important words of a cluster can be identified by the weight values.

For understanding and interpretation of clustering results, a few keywords that can best represent the semantic topic are extracted from each cluster. Two methods are used to extract the representative words. The candidate words are first selected according to their weights calculated by our new algorithm. Then, the candidates are fed to the WordNet to identify the set of noun words and consolidate the synonymy and hyponymy words. Experimental results have shown that the clustering algorithm is superior to the other subspace clustering algorithms, such as *PROCLUS* and *HARP* and k-means type algorithm, e.g., *Bisecting-KMeans*. Furthermore, the word extraction method is effective in selection of the words to represent the topics of the clusters.

***Keywords*—** Subspace Clustering, Text Mining, Feature Weighting, Cluster Interpretation, Ontology

I. Introduction

TEXT data is ubiquitous on the Web, in enterprise information systems and in personal files. As the volume of text data increases at an astonishing speed, management and analysis of text data becomes unprecendentedly important. Text mining is being developed as an emerging technology to handle the increasing text data. Tools and systems are being developed by both traditional data mining tool vendors such as SAS's TextMiner[1] and Megaputer's TextAnalyst [2], and new comers such as UMN's gCLUTO [3] and THOMSON's RefViz [4].

Text clustering is one of the fundamental functions in text mining [5]. Clustering is to divide a collection of text documents into different category groups so that documents in the same category group describe the same topic such as classic music or Chinese history. There are many uses of clustering in real applications, for example, grouping the Web search results and categorizing digital documents. Unlike clustering structured data, clustering text data faces a number of new challenges. Among others, the volume of text data, dimensionality, sparsity and complex semantics are the most important ones. These characteristics of text data require clustering techniques to be scalable to large and high dimensional data, and able to handle sparsity and semantics.

Different from the structured data stored in relational databases, text data sources are either semi-structured, such as XML data or unstructured such as, free text. However, most existing clustering algorithms were designed for structured data. To apply them to text data, the original text formats have to be transformed into structured forms. The commonly used structured form for text data is Vector Space Model [6] in which individual text documents are represented as a set of vectors. In transformation of original source text data to the vector space model, a number of preprocessing steps are used, including filtering, stemming, term frequency calculation, term selection, etc. These preprocessing steps are very important because they could significantly affect the results of text clustering.

In this paper, we present a text clustering system for text mining. This system consists of six components: *preprocessing, VSM data model, clustering, postprocessing, visualization* and *ontology*. The *preprocessing* component provides necessary text data preprocessing functions as above mentioned. One of the innovative implementations in this system is the use of domain specific *ontology* [7] to interpret the clustering results.

In the *clustering* component, we use a new subspace clustering algorithm that is based on extensions of the k-means algorithm to cluster large text data [8]. In this algorithm, we add a new step in the k-means clustering process to automatically calculate the feature weights for each cluster so that the important features to form a cluster subspace can be identified by the weight values. This extension enables the k-means clustering algorithm to cluster high dimensional text data in subspaces of the keyword features, so that sparsity of text data can be effectively handled. Since the additional step does not increase the number of iterations, the performance of the k-means clustering process is preserved.

One difficulty in clustering large text data is to understand and interpret the clustering results. If the number of text documents was small, a cluster could be understood by looking into the content of all documents in different clusters. If the number of text documents is large, reading the content of all documents becomes infeasible. Instead of looking into the document content, we can extract a few keywords from each cluster that can best represent the semantic topic of the cluster. However, the keyword extraction itself is a research challenge in text clustering. In the *postprocessing* component, we use two methods to extract the representative keywords from each

L. Jing is with E-Business Technology Institute & Department of Mathematics, The University of Hong Kong, Pokfulam Road, Hong Kong. E-mail: lpjing@eti.hku.hk

Michael K. Ng is with the Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong. E-mail: mng@math.hkbu.edu.hk

Xinhua Yang is with Software Technology Institute, Dalian Jiaotong University, DaLian, China. E-mail: yangxh@djtu.edu.cn

Joshua Z. Huang is with E-Business Technology Institute, The University of Hong Kong, Pokfulam Road, Hong Kong. E-mail: jhuang@eti.hku.hk

cluster. We first make use of the feature weights to identify the candidates of keywords for each cluster. Then, we use *ontology* (e.g, WordNet [9]) to identify the word functions. The final selection of keywords is based on the results of the two methods, for instance, the words with high weight value, high frequency and noun function.

The *visualization* component visualizes the clusters as semantic networks. Each node represents a cluster whose topic is described in a small set of keywords. The edge indicates the relationship between two clusters. We provide interactive operations on the network to allow the user to interactively adjust the clustering results by changing the keyword set of a cluster or moving misclassified documents from one cluster to another. After the manual adjustment, the cluster algorithm can be re-executed to update the result.

In this paper, we present the clustering results of the subspace clustering algorithm in comparison with the results of *Bisecting-KMeans* and two typical existing subspace clustering approaches *PROCLUS* [10] and *HARP* [11] from the 20-Newsgroups real-world text data set [12] and show the improvement in clustering accuracy and scalability of clustering process. We also show the effectiveness of use of multiple methods in selection of the keywords to represent the topics of the clusters.

The paper is organized as follows: Section II presents related work. Section III gives a brief description about the architecture of the text clustering system. Section IV describes the k-means type subspace clustering algorithm. The methods of using ontology (WordNet) to extract keywords are presented in Section V. Section VI presents experiment results to show both clustering performance and visualization. Some conclusions are drawn in Section VII.

II. Related Work

Clustering is an important topic in many disciplines. Three recent surveys on clustering [13], [14], [15] give a comprehensive summary of different clustering algorithms and applications. When applying to text data, these existing clustering methods face two big challenges, clustering of high dimensional and sparse text data and effective presentation of clustering results for easy interpretation and understanding. This work is motivated to solve these two problems by adopting a new subspace clustering method for text data and using WordNet [9] to extract few representative keywords for presentation of clusters.

In text clustering, a set of documents are represented in a matrix where each row vector $< t_1, t_2, \cdots, t_n >$ represents a document and each column represents a term or word in the vocabulary of the document set. Clustering algorithms, such as the *Standard KMeans* [16] and its variations [17], [18], as well as the hierarchical clustering methods [19], [20], are used to cluster the matrix data. In many real applications, the matrix can be very large because of the large vocabulary and the number of documents. If the set of documents to be clustered contains many different categories of documents, the matrix can be very sparse. Most existing clustering algorithms are not effective in clustering high dimensional sparse data because these algorithms cluster data on the full space while clusters in sparse data often exist in subspaces. This situation makes scalable subspace clustering methods [15], [13] good candidates for text clustering.

PROCLUS [10] and its variant ORCLUS [21] are typical among the subspace clustering algorithms. They are based on the traditional k-medoids approach, with a goal of minimizing the average within-cluster dispersion. The limitation of these methods is the determination of neighboring objects based on similarity calculations that involve all dimensions. If the number of relevant dimensions of each cluster is small, different members of a cluster may appear to be dissimilar when all dimensions are considered. In these methods, therefore, the neighboring objects of a medoid need not become from the same real cluster and the relevant dimensions suggested by them could be wrong. Meanwhile, *PROCLUS* is sensitive to the input parameters l, the average related dimensions of the clusters.

A hierarchical subspace clustering approach with automatic relevant dimension selection *HARP* was recently presented [11]. *HARP* makes use of the dynamic threshold loosening mechanism to find the relevant dimensions of the corresponding clusters. *HARP* is based on the assumption that two objects are likely to belong to the same cluster if they are very similar to each other along many dimensions. However, due to the hierarchical nature, the algorithm is intrinsically slow. Also, if the number of relevant dimensions per cluster is extremely low, the accuracy of *HARP* may drop as the basic assumption will become less valid due to the presence of large amount of noise values in the dataset.

The k-means clustering algorithm is known to be efficient in clustering large data sets. The recent development of the new k-means type algorithms with variable weighting ability enables the efficient k-means clustering process to discover clusters from subspaces that are identified by the weights of variables [22], [23]. The variable weights are calculated automatically in the clustering process with respect to the distributions of variables. The k-means type algorithms are effective in identifying noise variables in data and can be used for variable selection in data mining. We further extended the algorithm to effectively handle sparse data and successfully applied the new algorithm *FW-KMeans* [8] to large text data.

After clustering, the results are often visualized by showing the inter- and intra-relationships between objects within and across clusters. Techniques include projections [24] and 3D dendrograms [25]. These general techniques do not use the semantics of text. In text clustering, using keyword vectors to represent clusters was reported in [26]. A keyword vector was obtained from a word cluster that was separately generated. However, the concept vector for each cluster was obtained by associating with a word cluster that was separately generated. This process potentially affects the running time and complexity. Besides, the clustering approach used is the standard spherical k-means algorithm. Another interesting method for generating word clusters was given in [27]. In this method, word clusters can be generated by calculating the minimum loss of the mutual information between the words and the class labels. Word clusters can also be created using

statistical methods without considering semantics. In our work, we use an ontology-based method in combination with the feature weights and Zipf's law to extract keywords for cluster presentation.

III. Architecture of the Text Clustering System

The text clustering system consists of six components: preprocessing, VSM data model, clustering, postprocessing, visualization, and ontology. Fig.1 shows the architecture of the system. The functionality of each component is described as follows:

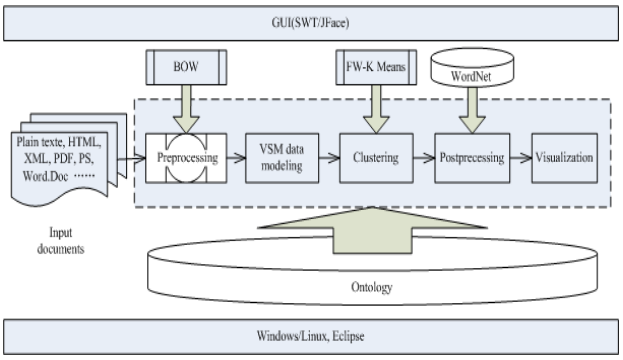


Fig. 1. architecture of the text clustering system

- ***Preprocessing:*** The preprocessing component provides functions to transform real text data in different formats into vector representations in the VSM data model, so that a clustering algorithm can be applied. The functions include parsers to parse real text data in plain text, HTML, XML, PDF, PS, Word into a set of terms, stop word removal, word stemming, term selection and term scoring such as $tfidf$. Currently, BOW toolkit [28] is used to implement these preprocessing functions.
- ***VSM data model:*** The VSM data model is used to represent text data as vectors to which clustering algorithms can be applied.
- ***Clustering:*** This component uses a k-means type subspace clustering method to perform the clustering process in the document vector space generated by the preprocessing component. It finds the topics (i.e., clusters) underlying the documents set and identifies the important words for each topic.
- ***Postprocessing:*** This component uses an electronic lexical database: WordNet to extract a few representative words (generally, from three to ten words) from the keyword candidates that are selected in the clustering component for each cluster. In addition, it extracts the common keywords between the clusters to show their relationships.
- ***Visualization:*** This component visualizes the semantics of each cluster with their own keywords and describes the relationship between the clusters with their common keywords.
- ***Ontology:*** The ontology component manages ontologies from different domains. These domain specific ontologies are used in other components to postprocess clustering results for interpretation and visualization.

In the following sections, we will present a new k-means type subspace clustering method for clustering text data and a method to use ontology WordNet to extract keywords for representing clusters.

IV. k-means Type Subspace Clustering

In this section, we present the k-means type subspace clustering algorithm *FW-KMeans* in detail. This algorithm is the core of the clustering component in the text clustering system.

A. Subspace Clustering

Let a set of text documents be represented as a set of vectors $\mathbf{X} = \{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n\}$. Each vector $\mathcal{X}_j$ is characterized by a set of *m* terms $(t_1, t_2, \ldots, t_m)$. Here, the terms can be words, phrases and high level concepts appearing in the documents. m is the total number of unique terms in all documents which form the vocabulary of these documents. This representation of text data is called vector space model (VSM). The terms are referred to as features. Let $\mathbf{X}$ be a set of documents that contain several categories. Each category of documents is characterized by a subset of terms in the vocabulary that corresponds to a subset of features in the vector space.

A simple illustration of text data in VSM is given in Table 1. Here, x_j represents the jth document vector; t_i represents the ith term; each cell in the table is the frequency that term t_i occurs in x_j. A zero cell means that the term does not appear in the related document. Documents x_0, x_1, x_2 belong to one category C_0, assuming *sport*, while x_3, x_4, x_5 belong to another category C_1, assuming *music*.

TABLE I
A Simple Illustration of Text Data in VSM

		t_0	t_1	t_2	t_3	t_4
C_0	x_0	1	2	3	0	2
	x_1	2	3	1	0	2
	x_2	3	1	2	0	2
C_1	x_3	0	0	1	3	2
	x_4	0	0	2	1	3
	x_5	0	0	3	2	1

Because these two categories are different, they are categorized by different subsets of terms. As shown in Table I, category C_0 is categorized by terms t_0, t_1, t_2 and t_4 while category C_1 by terms t_2, t_3 and t_4. In the meantime, terms play different roles on identifying categories or clusters. For instance, the same frequency of t_4 appears in every document of category C_0, hence, t_4 should be more important than other terms in identifying category C_0. The subspace clustering k-means algorithm to be discussed in the next subsections is able to distinguish the roles of different terms.

B. k-means with Feature Weighting

In clustering data, the k-means algorithm treats all features equally. To enable the k-means algorithm to identify differences of features in forming clusters, the basic k-means algorithm can be extended to allow calculation of a weight for each feature [23], [22] so the importance of a feature can be identified by the weight value. Mathematically, this new feature weighting k-means algorithm minimizes the following objective function:

$$F(W,Z,\Lambda)=\sum_{l=1}^{k}\sum_{j=1}^{n}\sum_{i=1}^{m}w_{l,j}\lambda_{l,i}^{\beta}d(z_{l,i},x_{j,i}) \quad (1)$$

subject to

$$\begin{cases} \sum\limits_{l=1}^{k} w_{l,j}=1, & 1\le j\le n \\ w_{l,j}\in\{0,1\}, & 1\le j\le n, \quad 1\le l\le k \\ \sum\limits_{i=1}^{m}\lambda_{l,i}=1, & 0\le\lambda_{l,i}\le 1, \quad 1\le l\le k \end{cases} \quad (2)$$

where $k(\le n)$ is the number of clusters; β (>1) is an exponent [23]; $W=[w_{l,j}]$ is a $k\times n$ integer matrix; $Z=[Z_1,Z_2,\ldots,Z_k]\in\mathbf{R^{k\times m}}$ are the k cluster centers; $\Lambda=(\Lambda_1,\Lambda_2,\ldots,\Lambda_k)$ is the set of weight vectors for all clusters in which each $\Lambda_l=(\lambda_{l,1},\lambda_{l,2},\ldots,\lambda_{l,m})$ is a vector of weights for m features of the lth cluster; $d(z_{l,i},x_{j,i})$ (≥ 0) is a distance or dissimilarity measure between the jth document and the center of the lth cluster on the ith feature. In text clustering, we use the Euclidean distance $d(x_{j,i},z_{l,i})=(x_{j,i}-z_{l,i})^2$ because the frequencies of terms are numeric values.

The unknowns W and Z are solved in the same way as the standard k-Means algorithm [29]. Each feature weight $\lambda_{l,i}$ is solved with the Lagrange multiplier technique and obtained by:

$$\lambda_{l,i}=\frac{1}{\sum_{t=1}^{m}\left[\frac{\sum_{j=1}^{n}\tilde{w}_{l,j}d(\tilde{z}_{l,i},x_{j,i})}{\sum_{j=1}^{n}\tilde{w}_{l,j}d(\tilde{z}_{l,t},x_{j,t})}\right]^{1/(\beta-1)}} \quad (3)$$

This algorithm finds the clusters from different subspaces by automatically calculating weights for each features in every cluster, higher weight for important feature, otherwise the feature will be set a lower weight. However, in large, high dimensional and sparse data, there is one special case that a cluster can not contain all of the features, that means there is one or more than one features do not appear in any object of one cluster, such as terms t_3 and t_4 in C_0 of Table 1. This situation will cause zero dispersion, i.e., the term $\sum_{j=1}^{n}\tilde{w}_{l,j}d(\tilde{z}_{l,i},x_{j,i})$ in Eq.(3) will be zero, and then the relative λ_{li} will become infinite so the objective function Eq.(1) cannot be minimized properly.

C. Handle Sparsity of Text Data

In this section, we propose a new subspace clustering method called by *FW-KMeans*, that effectively solves the sparsity problem mentioned above. The new approach uses a modified objective function (1) by adding a constant σ to the distance function as:

$$F_1(W,Z,\Lambda)=\sum_{l=1}^{k}\sum_{j=1}^{n}\sum_{i=1}^{m}w_{l,j}\lambda_{l,i}^{\beta}[d(z_{l,i},x_{j,i})+\sigma] \quad (4)$$

With the introduction of σ, the dispersion of a feature in a cluster can never be zero so that the objective function (4) can be minimized properly. The value of parameter σ will affect the feature weighting process. If σ is much larger than $d(z_{l,i},x_{j,i})$, the weights will be dominated by σ and $\lambda_{l,i}$ will approach to $\frac{1}{m}$. This will make the clustering process back to the standard k-means. If σ is too small, then the gap of the weights between the zero dispersion features and other important features will be big, therefore, undermining the importance of other features. To balance, we calculate σ as the average dispersion of the entire data set for all features as follows:

$$\sigma=\frac{\sum_{j=1}^{\hat{n}}\sum_{i=1}^{m}d(x_{j,i},o_i)}{\hat{n}\cdot m} \quad (5)$$

where o_i is the mean feature value of the entire data set. In practice we use a sample instead of the entire data set to calculate σ. (5% sample can be used according to the sampling theory [30]). $\hat{n}$ is the number of documents in the sample. Experimental results in Section 6 have shown that this selection of σ is reasonable to produce satisfactory clustering results and identify important features of clusters.

Eq.(4) can be minimized by iteratively solving the following three problems:

Fix $\tilde{Z}$ and $\tilde{\Lambda}$, compute the partition matrix W by

$$\begin{cases} w_{l,j}=1 & \text{if} \quad \sum\limits_{i=1}^{m}\lambda_{l,i}^{\beta}(d(z_{l,i},x_{j,i})+\sigma) \\ & \quad \le \sum\limits_{i=1}^{m}\lambda_{h,i}^{\beta}(d(z_{h,i},x_{j,i})+\sigma) \\ & \quad \text{for } 1\le h\le k \\ w_{l,j}=0 & \text{otherwise} \end{cases} \quad (6)$$

Similarly, the optimal solution W is not unique, and $w_{l,j}=1$ is arbitrarily assigned to the first minimizing index l, and the remaining entries of the column are put to zero. And $w_{l,j}=1$ means that the jth document belongs to the lth cluster because they have the smallest distance.

Fix $\tilde{W}$ and $\tilde{\Lambda}$, and compute the set of cluster center vectors Z by

$$z_{l,i}=\frac{\sum\limits_{j=1}^{n}w_{l,j}\,x_{j,i}}{\sum\limits_{j=1}^{n}w_{l,j}} \qquad \text{for } 1\le l\le k \text{ and } 1\le i\le m \quad (7)$$

Fix $\tilde{W}$ and $\tilde{Z}$, compute the feature weighting matrix Λ by

$$\lambda_{l,i}=\frac{1}{\sum_{t=1}^{m}\left[\frac{\sum_{j=1}^{n}\tilde{w}_{l,j}[d(\tilde{z}_{l,i},x_{j,i})+\sigma]}{\sum_{j=1}^{n}\tilde{w}_{l,j}[d(\tilde{z}_{l,t},x_{j,t})+\sigma]}\right]^{1/(\beta-1)}} \quad (8)$$

These three solutions define the feature weighting k-means algorithm *FW-KMeans* as follows:

1) Choose initial cluster centers $Z^{(0)}\in\mathbf{R^{k\times m}}$ and set $\Lambda^{(0)}$ with all entries equal to $1/m$. Set $h=0$.
2) Use Eq.(6) to calculate $W^{(h+1)}$. If $F(W^{(h+1)},Z^h,\Lambda^h)=F(W^h,Z^h,\Lambda^h)$, then stop; otherwise, goto step 3.

3) Use Eq.(7) to calculate $Z^{(h+1)}$. If $F(W^{(h+1)}, Z^{(h+1)}, \Lambda^h) = F(W^{(h+1)}, Z^h, \Lambda^h)$, then stop; otherwise, goto step 4.
4) Use Eq.(8) to calculate $\Lambda^{(h+1)}$. If $F(W^{(h+1)}, Z^{(h+1)}, \Lambda^{(h+1)}) = F(W^{(h+1)}, Z^{(h+1)}, \Lambda^h)$, then stop; otherwise, set $h = h + 1$ and goto step 2.

D. Algorithm Analysis

Using a similar approach in [31], we can show that the new algorithm converges to a local minimal solution in a finite number of iterations. We note that there are a finite number of possible partitions W. Each possible partition W appears at most once by the algorithm. Assume that $W^{h_1} = W^{h_2}$ where $h_1 \neq h_2$. We note that given W^h, we can compute the minimizer Z^h which is independent of Λ^h according to Eq.(7). For W^{h_1} and W^{h_2}, we have the minimizers Z^{h_1} and Z^{h_2} respectively. It is clear that $Z^{h_1} = Z^{h_2}$ since $W^{h_1} = W^{h_2}$. Using W^{h_1} and Z^{h_1}, and W^{h_2} and Z^{h_2}, we can compute the minimizers Λ^{h_1} and Λ^{h_2} respectively (Step 4) according to Eq.(3). Again, $\Lambda^{h_1} = \Lambda^{h_2}$. Therefore, we have

$$F_1(W^{h_1}, Z^{h_1}, \Lambda^{h_1}) = F_1(W^{h_2}, Z^{h_2}, \Lambda^{h_2}).$$

However, the sequence $F_1(\cdot, \cdot, \cdot)$ generated by the algorithm is strictly decreasing. Hence the result follows.

Since the *FW-KMeans* algorithm is an extension to the k-means algorithm by adding a new step to calculate the feature weights in the iterative process, it does not seriously affect the scalability of the k-means type algorithms in clustering large data. The run-time complexity of this algorithm can be analyzed as follows. In this algorithm, there are essentially three major computation steps:

- ***Partitioning Documents:*** After initialization of the feature weights and cluster centers in Step 1, A cluster label will be assigned to each document. This process simply compares the summation $\sum_{i=1}^{m} \lambda_{l,i}^{\beta}(d(z_{l,i}, x_{j,i}) + \sigma)$, for each document in all k clusters. Thus, the complexity for this step is $O(mnk)$.
- ***Updating Cluster Centers:*** With the partition matrix W, the task of updating cluster centers is to find the means of the features in the documents which belong to the same cluster. Thus, for k clusters, the run-time complexity for this step is $O(mnk)$.
- ***Calculating Feature Weights:*** The last phase of this algorithm is to calculate the feature weights for all clusters based on the partition matrix W. In this step, we only go through the whole data set once to update the feature weights. The runtime complexity of this step is also $O(mnk)$.

Therefore, the total runtime complexity of this algorithm is $O(hmnk)$, where h is the total number of iterations.

As for the storage, we need $O(2mk)$ space to hold the cluster centers Z and the feature weighting matrix Λ and $O(n(1 + \bar{m}))$ space to store the set of n documents and their cluster labels, where $\bar{m}$ is the average number of terms in each document and $\bar{m} << m$, because we only store the nonzero entries as shown in Table 1 in order to save the memory consumption.

V. Clustering Evaluation and Interpretation

Evaluation and interpretation of clustering results are two important steps in clustering text data. This section presents the methods used for these two purposes. In clustering evaluation, if the number of text documents to be clustered is small, the clustering results can be easily evaluated by reading the contents of the documents in each cluster. However, this approach becomes infeasible when the number is very large. Instead, we can use samples to evaluate different settings of the clustering algorithm and apply the best setting to the large data set.

For cluster interpretation, the feature weights, the Zipf's law [32] and WordNet are used to jointly identify a few words from each cluster to represent clusters and their relationships.

A. Clustering Evaluation

The class labels of the data used in the experiments are actually known, therefore we can adopt the existing measures, *Accuracy*, *Entropy*, *F1* score ($FScore$) [20] and the normalized mutual information (*NMI*) [33], to objectively assess the clustering performance. We note that the labels are not used in the clustering process.

These four measures are defined as follows. Given a set of text documents X in k categories C_1,C_2,...,C_h,...C_k, we use a clustering algorithm to cluster it into k clusters S_1,S_2,...,S_l,...,S_k. Let n_h, n_l be the numbers of documents in category C_h and cluster S_l respectively, $n_{h,l}$ be the number of documents appearing in both category C_h and cluster S_l, and n be the total number of documents in X. A confusion matrix can be defined as Table II.

TABLE II
Confusion Matrix

	S_0	S_1	S_2	S_3
C_0	$n_{0,0}$	$n_{0,1}$	$n_{0,2}$	$n_{0,3}$
C_1	$n_{1,0}$	$n_{1,1}$	$n_{1,2}$	$n_{1,3}$
C_2	$n_{2,0}$	$n_{2,1}$	$n_{2,2}$	$n_{2,3}$
C_3	$n_{3,0}$	$n_{3,1}$	$n_{3,2}$	$n_{3,3}$

From the confusion matrix, the accuracy of a clustering is calculated by

$$Accuracy = \frac{\sum_{l=1}^{k} n_{l,l}}{n} \tag{9}$$

The entropy is defined as

$$Entropy = \sum_{l=1}^{k} \frac{n_l}{n} \left(-\frac{1}{\log k} \sum_{h=1}^{k} \frac{n_{h,l}}{n_l} \cdot \log \frac{n_{h,l}}{n_l} \right) \tag{10}$$

The $FScore$ is defined as

$$FScore = \sum_{h=1}^{k} \frac{n_h}{n} \cdot \max_{1 \leq l \leq k} \left\{ \frac{2 \cdot n_{h,l}/n_h \cdot n_{h,l}/n_l}{n_{h,l}/n_h + n_{h,l}/n_l} \right\} \tag{11}$$

The normalized mutual information (*NMI*) is defined as

$$NMI = \frac{\sum_{h,l} n_{h,l} log \left(\frac{n \cdot n_{h,l}}{n_h n_l} \right)}{\sqrt{(\sum_h n_h log \frac{n_h}{n})(\sum_l n_l log \frac{n_l}{n})}} \tag{12}$$

In Eq.(9), the term $n_{l,l}$ is the number of documents occurring in both lth cluster and its corresponding category. For other three evaluation measures, the index of a cluster and its corresponding category need not match with each other.

In these measures, the *Entropy* is more comprehensive than accuracy because instead of just considering the number of objects 'in' and 'not in' the dominant class, it takes the entire

distribution into account. However, the *Entropy* measure is biased towards small clusters (e.g., a set of singletons is always considered perfect).

The *FScore* measure combines precision and recall into a single number with equal weights. Unlike entropy, *FScore* is not biased towards small clusters. In fact, it favors coarser clusterings.

The normalized mutual information (*NMI*) avoids the bias toward smaller clusters. Singletons are severely discounted compared to the best clustering. However, *NMI* for the best clustering is smaller than 1 unless all categories have equal prior probabilities. Thus, *NMI* provides an unbiased and conservative measure as compared to $FScore$ and entropy.

In these four clustering measures, *Accuracy*, *FScore* and *NMI* rank the clustering quality from 0 (worst) to 1 (best), while the *Entropy* measure from 1 (worst) to 0 (best).

B. Key Word Extraction for Cluster Interpretation

After clusters are generated by the clustering algorithm, a few keywords can be extracted from each cluster to represent the cluster's semantic category. Since the weights reflect the importance of the terms in forming a cluster, we first select the keyword candidates according to the weight values. Given a weight threshold, we identify keyword candidates whose weights are greater than the threshold for each cluster (about 17% of the vocabulary). After obtaining the candidate key words, we use Zipf's law to delete the candidate words that occur in less than 15% of the total documents in the cluster [32]. These words cannot represent the semantic category of the majority documents of the cluster.

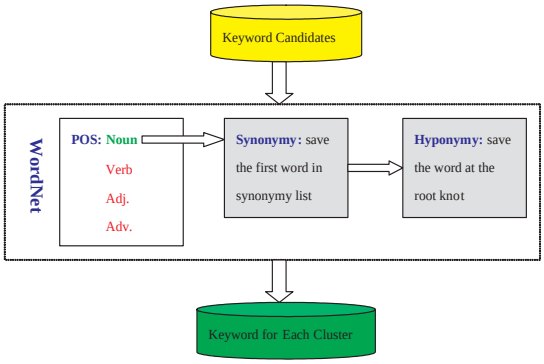


Fig. 2. extract keywords with WordNet.

The keyword candidates after Zipf's law filtering are fed to the WordNet [9] for the final selection of a small set of keywords for each cluster. WordNet is an electronic lexical database that describes the function of each English word, such as NOUN, VERB, ADJECTIVE and ADVERB, and the semantic relationships between the words, such as synonyms and hyponyms. We use WordNet to perform two tasks, (1) selecting the noun candidates and (2) consolidating the sets of the synonym and hyponym words. As shown in Fig.2, for each keyword candidate, we use WordNet to check whether it is a noun or not. If it is not a noun word, we remove it from the candidate list. For the list of candidate nouns, we identify the sets of synonyms and hyponyms. If the words are synonyms, we consolidate them by retaining one word and deleting others for each synonymy set, as shown in Fig.3(a). If the words are hyponymy, we choose the general word to represent the rest. For example in Fig.3(b), we retain the word arm to represent weapon hyponyms. The algorithm for keywords extraction with the aid of Ontology (WordNet) is described as follows.

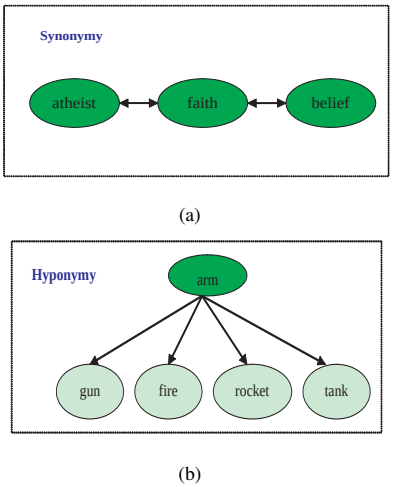


Fig. 3. example for extracting synonymy and hyponymy keywords.

Algorithm — (*KeyWExtractor*)

1) Input the features with weights obtained by our clustering approach *FW-KMeans* and set the threshold θ for feature weights;
2) For each cluster C_l $(1 \leq l \leq k)$ {*find keyword candidates for every cluster*}
   a) IF $\lambda_{l,i} < \theta$, remove the i^{th} term from the corresponding cluster C_l;
   b) ELSE assign the i^{th} term as the keyword candidate of the cluster C_l;
3) Extract keywords for each cluster with the aid of WordNet
   a) For each keyword candidate kwc_j {*check word* kwc_j *is noun or not*}
      i) IF $ss\_type(kwc_j)! = n$
      remove it from the keywords set of the clusters;
      {*n represents for noun, v for verb, a for adjective and r for adverb*}
      ii) ELSE keep the word;
   b) For all keyword candidates $kwc_{l,j}$ in each cluster C_l $(1 \leq l \leq k)$ {*check there are synonyms or hypernyms in the keyword set for each cluster* }
      i) IF $kwc_{l,j_1} \in hype(kwc_{l,j_2}, n)$ or $kwc_{l,j_1} \in syne(kwc_{l,j_2}, n)$
      remove $kwc_{l,j2}$ from the keyword set of the cluster C_l;
      { $hype(t, n)$ *return the hypernym for noun word* t*;* $syne(t, n)$ *returns the synonyms and immediate hypernyms of the noun word* t}
      ii) ELSE keep the word;
   c) Output the keywords for each cluster.

Finally, we analyze the keywords in different clusters and identify the keywords which occur in more than one cluster. In doing so, the related topics will be easily partitioned. These common words can also identify the relationships between clusters. The keywords for each cluster and the common words between clusters are used to represent the semantics

and relationships of clusters when visualizing the clustering results.

VI. Experiments

In this section, we present some experiment results to demonstrate the performance of the *FW-KMeans* algorithm with comparison of *Bisecting-KMeans* and two typical subspace clustering methods *PROCLUS* and *HARP*, and also show the effectiveness of the word extraction methods. The experiments were conducted with real-world text data sets on a machine with a 3.2G CPU and 2G RAM.

A. Text Datasets

In conducting the experiments, 21 datasets with different characteristics in sparsity, size, dimensionality and category distribution were previously built from the well known *20-Newsgroups* collection. This dataset is a collection of 20,000messages, collected from 20 different newsgroups. The documents in each category of our experimental datasets are chosen at random from the corresponding group.

Table III lists the first 6 datasets that were used to test the clustering quality. The source column gives the categories of each dataset. Column n_d indicates the number of documents in each category. Data sets A2 and A4 contain categories of quite different topics while datasets B2 and B4 consist of categories of similar topics. Obviously, clustering B2 and B4 is harder than A2 and A4 because there are more overlapping words in B2 and B4. Datasets A4-U and B4-U contain unbalanced numbers of documents in different categories.

Table IV lists other 15 datasets that were used to test the scalability of the algorithm. In the first group of datasets D_{1-6}, each dataset has 15905 documents in 20 categories. The number of terms (m) in these datasets increases from 500 to 2000. In the second group of datasets E_{1-4}, the number of categories in each dataset is fixed to 20 and the number of terms used to represent these datasets is 1100. The number of documents in these datasets increases from 2000 to 15905. These two sets of datasets cover all 20 major topics in the *20-Newsgroups* collection. In the last group of datasets F_{1-5}, the number of categories k in each dataset increases while the number of documents and the number of terms are fixed to 1500 and 500 respectively. The 12 categories in F_{1-5} are *alt.atheism*, *comp.graphics*, *talk.politics.guns*, *rec.autos*, *soc.religion.christian*, *misc.-forsale*, *sci.crypt*, *comp.sys.ibm.pc.hardware*, *rec.sport.basketball*, *sci.space*, *comp.os.windows*, and *talk.politics.mideast*.

The documents in these 21 datasets were first processed with the component of our system, preprocessing. Firstly, the system striped each news message from the e-mail header and special tags, eliminated stop words and stemmed words to their root form. Next, words were sorted based on their inverse document frequency (IDF). Finally, the feature was removed by checking whether its IDF value was in the range of threshold.

B. Experimental Results

In this subsection, we first present the experimental results of clustering quality and scalability of the clustering algorithm *FW-KMeans*. Then, we show the results of keyword extraction. In conducting these experiments, we used 5% of sample documents to choose the initial centers as the farthest *k* points between two categories in the sample data [34]. We set proper parameters for different algorithms so that they could get better results. β is 1.5 for *FW-KMeans* and l is set a value in [$10\%m$, $80\%m$] (m is the number of features) for *PROCLUS*. For *Bisecting-KMeans*, *FW-KMeans*, *HARP*, we run each of them ten times and then reported the average values. For *PROCLUS*, we assigned different values for the parameter l (8 values), and run the algorithm ten times for each l value, then computed the average result for each l, finally reported the best results in all l cases. Here, we use *Bisecting-KMeans* to compare with our algorithm because it is one of the document clustering methods with the best performance [18], and *cosine* similarity is the similarity measure between objects in *Bisecting-KMeans*.

1) Clustering Quality: Four clustering algorithms, *FW-KMeans*, *Bisecting-KMeans*, *PROCLUS* and *HARP* were applied to the 6 datasets in Table III. The clustering results evaluated with the four evaluation measures in Subsection 5.1 are given in Table V. The four figures in each cell represent the values of Accuracy, Entropy, Fscore and NMI respectively. Each accuracy value was calculated as the average of accuracy values of five runs on the same dataset.

From Table V, we can see that the *FW-KMeans* performed best in most cases. The performance of *FW-KMeans* is always better than the other subspace clustering methods *PROCLUS* and *HARP* for all datasets. This result is not hard to explain. *PROCLUS* just used neighboring objects to determine the importance of the relevant features, however, this measure would result in wrong decision if the neighbors were not in the same real cluster. However, our method, *FW-KMeans*, calculates the features weights depending on the data distribution in one cluster. Meanwhile, *HARP* did not get good performance because its assumption became less valid for sparse text data.

The *Bisecting-KMeans* performed slightly better than the *FW-KMeans* on A2 and A4 because the categories in these datasets are quite different, meaning that the sets of terms in different clusters do not overlap much. For datasets B2 and B4 with more overlapping terms, the *FW-KMeans* performed much better.

TABLE VI
Confusion Matrix of B4-U by *Bisecting-KMeans*

	S_0	S_1	S_2	S_3
C_0	21	2	70	27
C_1	30	0	11	59
C_2	1	53	1	4
C_3	0	10	4	6

TABLE III
DATASETS FOR TESTING THE CLUSTERING QUALITY

DataSet	Source	n_d	DataSet	Source	n_d
A2	alt.atheism	100	B2	talk.politics.mideast	100
	comp.graphics	100		talk.politics.misc	100
A4	comp.graphics	100	B4	comp.graphics	100
	rec.sport.baseball	100		comp.os.ms-windows	100
	sci.space	100		rec.autos	100
	talk.politics.mideast	100		sci.electronics	100
A4-U	comp.graphics	120	B4-U	comp.graphics	120
	rec.sport.baseball	100		comp.os.ms-windows	100
	sci.space	59		rec.autos	59
	talk.politics.mideast	20		sci.electronics	20

TABLE IV
DATASETS FOR TESTING THE SCALABILITY OF THE CLUSTERING ALGORITHM

Dataset	n	**m**	k	Dataset	**n**	m	k	Dataset	n	m	**k**
D_1	15905	500	20	E_1	2000	1100	20	F_1	1500	500	3
D_2	15905	800	20	E_2	4000	1100	20	F_2	1500	500	5
D_3	15905	1100	20	E_3	8000	1100	20	F_3	1500	500	7
D_4	15905	1300	20	E_4	15905	1100	20	F_4	1500	500	10
D_5	15905	1700	20					F_5	1500	500	12
D_6	15905	2000	20								

TABLE V
CLUSTERING RESULTS OF *FW-KMeans*, *Bisecting-KMeans*, *PROCLUS* AND *HARP*. (BOLD-FACE NUMBERS INDICATE THE BEST EVALUATION RESULT IN THE FOUR ALGORITHMS)

	A2	B2	A4	B4	A4-U	B4-U
FW-KMeans	0.96	**0.905**	0.8975	**0.8621**	**0.9591**	**0.9197**
	0.2057	**0.4014**	0.2509	**0.3574**	**0.1513**	**0.2314**
	0.9599	**0.9043**	0.9003	**0.8631**	**0.9591**	**0.9205**
	0.7961	**0.6050**	0.7554	**0.6467**	**0.8480**	**0.7385**
Bisecting-KMeans	**0.965**	0.88	**0.9375**	0.7017	0.8954	0.6087
	0.2146	0.5294	**0.1919**	0.6195	0.2830	0.5357
	0.9650	0.8800	**0.9376**	0.7049	0.8961	0.6586
	0.7857	0.4706	**0.8083**	0.3822	0.7126	0.3793
PROCLUS	0.6884	0.6500	0.5725	0.4322	0.6167	0.5302
	0.5254	0.8395	0.5548	0.7291	0.7342	0.5758
	0.7190	0.6604	0.6450	0.4911	0.5239	0.5739
	0.2334	0.0789	0.2909	0.0791	0.1867	0.1684
HARP	0.8894	0.6000	0.5000	0.3769	0.6000	0.4228
	0.5016	0.9562	0.7671	0.8933	0.8389	0.9535
	0.8894	0.6020	0.5073	0.3840	0.4819	0.3364
	0.4984	0.0299	0.2023	0.0538	0.1688	0.0250

TABLE VII
CONFUSION MATRIX OF B4-U BY *FW-KMeans*

	S_0	S_1	S_2	S_3
C_0	109	9	0	2
C_1	3	95	1	1
C_2	0	3	54	2
C_3	2	1	0	17

For unbalanced datasets A4-U and B4-U, *FW-KMeans* performed reasonably well while the performance of *Bisecting-KMeans* clearly deteriorated. This was because the *Bisecting-KMeans* chose a branch to split at each step. Usually, the largest cluster was chosen. This resulted in artificial division of some inherent large clusters in the early stage so the mistake could not be corrected in the later stage. This can be observed by the confusion matrix in Table VI produced from B4-U. The large clusters C_0 and C_1 were divided into separate clusters by the *Bisecting-KMeans*. However, as shown in Table VII, the *FW-KMeans* algorithm recovered them more accurately.

2) Scalability: The fourteen datasets in Table IV were used to evaluate the scalability of the *FW-KMeans* algorithm. The experiment results demonstrated that the algorithm was scalable to the number of documents, the number of terms and the number of clusters. The algorithm also converged quickly after a few iterations on these test datasets.

Fig.4 shows the results on datasets D_{1-6}. We can see that both run-time and clustering accuracy increased linearly as the number of terms increased. This was in line with our analysis

of the linear computational complexity $O(h\mathbf{m}nk)$. The accuracy increased as the number of terms increased, because more terms held more information about the documents.

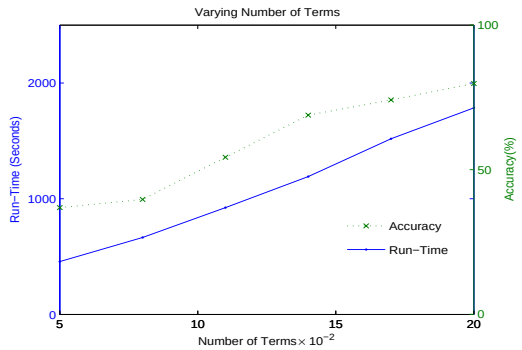


Fig. 4. run-time and accuracy for different numbers of terms.

Fig.5 shows the results on datasets E_{1-4}. In these datasets, the number of terms was fixed to 1100 and the number of clusters to 20. We can see that the run-time increased linearly as the number of documents increased. This was conformable to the analysis of the linear computational complexity $O(hm\mathbf{n}k)$. The clustering accuracy decreased as the number of documents increased. The reason was that the fixed number of terms might not well represent the whole documents set including newly added documents, which implicitly reduced the information of the documents and compromised the clustering quality.

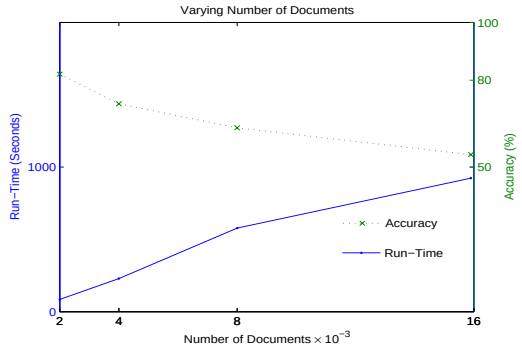


Fig. 5. run-time and accuracy for different numbers of documents.

Fig.6 shows the results on datasets F_{1-5}. In these datasets, the number of terms was fixed to 500 and the number of documents to 1500 while the number of clusters varied from 3 to 12. We can observe that the run-time increased linearly as the number of clusters increased. The fixed number of terms resulted in a slightly reduction of clustering accuracy because it implicitly reduced the information of the documents.

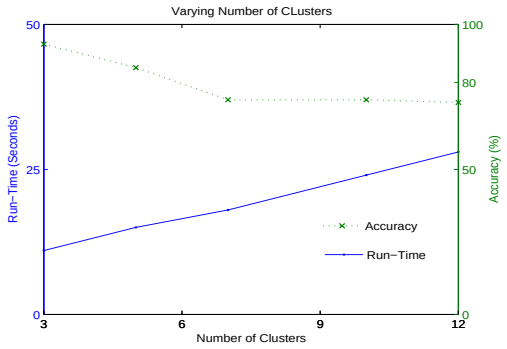


Fig. 6. run-time and accuracy for different numbers of clusters.

Fig.7 shows a typical convergency curve of the *FW-KMeans* algorithm produced from dataset F_3. The horizontal axis represents the number of iterations and the vertical axis represents the value of the objective function Eq.(4). Each point on the curve represents a partition generated by one iteration of the k-means clustering process. Starting from a set of initial cluster centers and a set of initial weights, the algorithm first converged after 2 iterations. A new set of weights $FW1$ were computed. Using $FW1$ as the initial weights and the current cluster centers, the k-means process restarted again. We can see that the objective function had a significant drop after the new weights $FW1$ were introduced. The k-means process converged again after 2 new iterations. Then, a set of new weights $FW2$ were computed. This process continued until the local minimal value of the objective function was reached. The final set of weights $FW4$ was obtained.

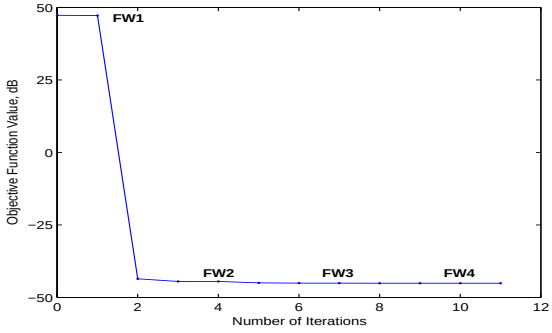


Fig. 7. convergence curve of the *FW-KMeans* algorithm. $dB = 10 \times log(value)$

The convergency curve indicates that the *FW-KMeans* algorithm will converge in a few iterations. For all fifteen datasets in Table IV, the maximal number of iterations was less than 15.

3) Keywords Extraction: After clusters were generated by the *FW-KMeans* algorithm, a few keywords were extracted from each cluster to represent the semantics of the cluster for the user. The three methods described in Section 5.2 were used in keyword extraction. First, all terms were sorted on the weights and divided into groups based on the weight intervals. Fig.8 shows a typical distribution of terms on weight intervals

for category *comp.graphics* obtained from dataset B4. The left table gives the number of words in each interval and the rightFig.plots the distribution of words. Before plotting, the terms that did not occur in the documents of this cluster were removed from the term list. Because only the terms with large weights were interested, the first two groups of terms were selected, which was about 25% of the total terms in this cluster.

word weights are divided into five intervals:

	weight intervals	word number
0~1:	(0,1e-08]	8
1~2:	(1e-08,1e-07]	280
2~3:	(1e-07,1e-06]	433
3~4:	(1e-06,1e-05]	188
4~5:	(1e-05,1)	32

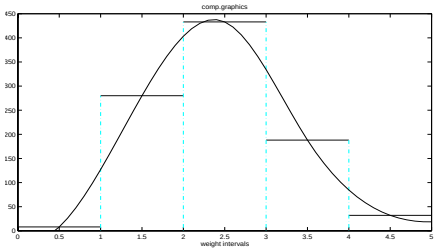


Fig. 8. distribution of word frequency on weights

Fig.9 shows the plots of selected terms or keywords of four clusters generated from dataset B4. The horizontal axis is the word index for the entire vocabulary and the vertical lines indicate the weights of selected keywords. We can see that the words with large weight values in each cluster do not overlap. This implies that the keywords selected according to their weights indeed can be used to separate documents in different clusters.

The right side of Fig.9 shows the ten noun words with the largest weights in each cluster. To relate these noun words to the categories of the clusters, we can find that some of these noun words closely represent the semantics of the clusters. For example, the category of the cluster in the top graph is computer graphics. The noun words *graphic, icon, image, color, point, gui* on the right side are all related to graphics. The same can be found in other clusters as well.

After selection with weights, Zipf's law [32] was used to check the candidate keywords against the documents in each cluster. The candidate words that occur in less than 15% of the total documents in the corresponding cluster were deleted from the candidate list because they were not useful in expressing the meaning of all documents in the cluster. After this removal, only about 20 to 35 words were left in each cluster. This step further reduced the size of the candidate keywords for cluster presentation.

The last step to select the final small set of keywords for each cluster was to use WordNet [9] to further delete some unnecessary words and retain the most important ones. Because the noun words can best represent the meaning of the clusters and should be retained, the first task of using WordNet was to identify noun words from the candidate words. Since WordNet provides semantic relationships between words, the semantic relationships were used to combine the synonymy or hyponymy words. The words in each group have the same meaning so that one word was retained. For instance, in dataset F_3, cluster 0 contains synonyms *atheism*, *atheist*, and *faith*, *belief*. Cluster 6 contains hyponymy words *weapon*, and *arm, fire, gun, rocket, tank*. In this case, the top word *weapon* in the hierarchical hyponymy structure was retained.

Finally, the words that occur in all clusters, such as the words $\{$*write, article, thing, people, human*$\}$ in F_3, were removed because they were not useful. For the words that occur in two clusters, they were retained to describe the relationship between the clusters. Table VIII shows the keywords in each cluster obtained from dataset F_3. We can see that each set of keywords is essentially correlated to only one of the seven topics listed in the column $Category_{Name}$. Here, n_l is the number of documents in the lth cluster.

C. Clustering Visualization

Clustering results are best shown through visualization. In this text clustering system, a visualization module was included to visualize the semantics of clusters and the relationships between clusters. The extracted keywords play an important role in presenting the semantics of clusters and the relationships between clusters. The user can interactively operate on the display to manually fine tune the results.

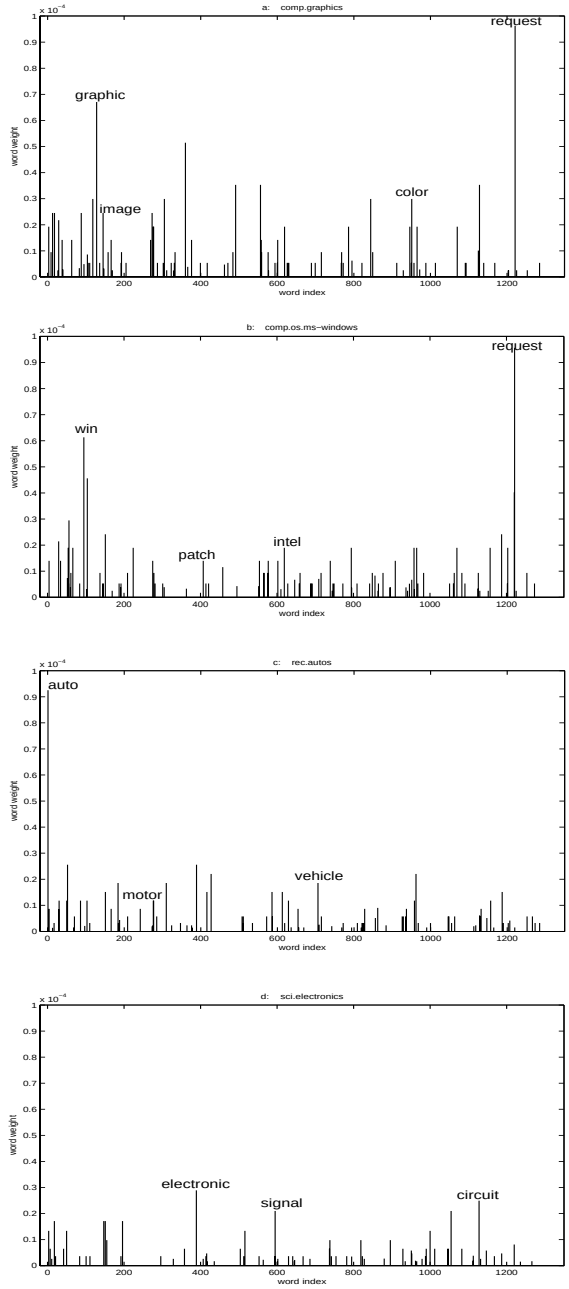
Fig.10 shows an example of graphic visualization of one clustering result produced from dataset E_4. Twenty clusters are displayed in the right panel. The squares with a small box in the top-left corner represent clusters. The number in the small box is the cluster ID. The keywords describing the cluster are displayed in the square. The squares without a small box represent the relationships between clusters. The keywords in the squares represent the semantic of the relationships. For example, the square with keywords $\{$*sport, game, score, team*$\}$ implies that cluster 9 and cluster 10 are overlapping because they share some common words. These two clusters could be combined into a high-level cluster.

The user can interact with the display to perform operations, such as viewing the complete list of keywords in a cluster, for example, the box of Clu-19, and creating a high level cluster by merging existing low level clusters. Through the interactive operations, the user can manually fine tune the clustering results, which is necessary in text clustering applications.

The left panel of this interface shows the functional operations of this text clustering system. These operations are used to form a text mining process with easy drag-and-drop operations.

VII. Conclusions

In this paper, we have presented a text clustering system with six components to tackle three challenging text clustering problems: large volume, high dimensionality and sparsity. The components of clustering, postprocessing and ontology are used to extract keywords for clustering visualization. With these three parts, the problem of complex semantics of text clustering can be effectively solved.



graphic	$6.70466e-05$
color	$2.98961e-05$
image	$2.45266e-05$
icon	$1.42196e-05$
laser	$9.52604e-06$
scope	$9.52604e-06$
point	$5.41076e-06$
sheet	$4.94495e-06$
plain	$3.21929e-06$
gui	$2.20811e-06$

win	$6.13444e-05$
intel	$2.14806e-05$
patch	$1.90001e-05$
logic	$1.15958e-05$
pc	$9.37718e-06$
buffer	$9.37718e-06$
demo	$8.34777e-06$
function	$5.32089e-06$
company	$5.32089e-06$
database	$3.91727e-06$

auto	$9.25063e-05$
vehicle	$1.8565e-05$
motor	$1.18095e-05$
driver	$9.01719e-06$
park	$8.57334e-06$
repair	$5.74717e-06$
mile	$4.15965e-06$
door	$3.23471e-06$
show	$3.21888e-06$
manufacture	$1.94154e-06$

electronic	$2.89103e-05$
circuit	$2.49422e-05$
signal	$2.10053e-05$
chip	$1.33768e-05$
volume	$9.80421e-06$
thread	$6.51865e-06$
charge	$3.67175e-06$
raster	$2.6509e-06$
science	$2.2915e-06$
technology	$1.91447e-06$

Fig. 9. weight distribution of keywords in four clusters from dataset *B4*. (a) cluster of category *Graphics*, (b) cluster of category *Windows*, (c) cluster of category *Autos*, (d) cluster of category *Electronics*.

TABLE VIII
REPRESENTATIVE WORDS EXTRACTED FROM SEVEN CLUSTERS OF DATASET F_3

$Cluster_{ID}$	$Category_{Name}$	n_l	Keywords
0	alt.atheism	232	{atheism, belief, moral}
1	comp.graphics	164	{graphic, image, software}
2	misc.forsale	366	{sale, price}
3	rec.autos	199	{engine, car, drive}
4	sci.crypt	187	{encrypt, key, chip, message}
5	soc.religion.christian	186	{christian, church, bible}
6	talk.politics.guns	166	{weapon, stratum, shoot}

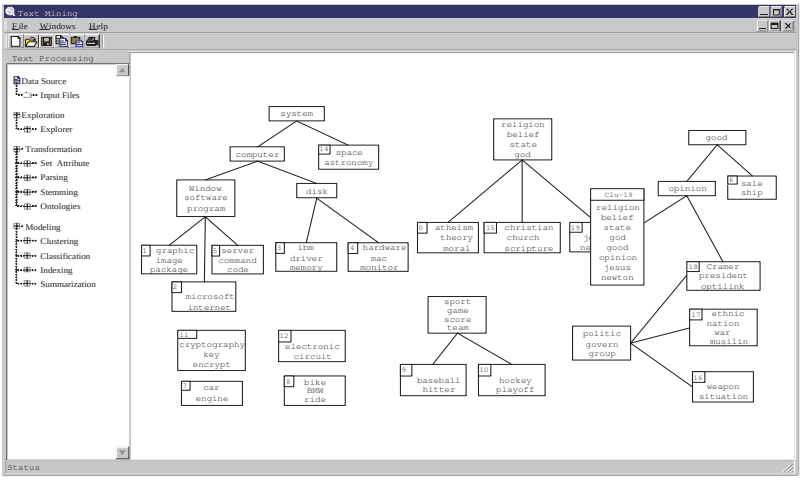


Fig. 10. visualization of clusters and relationships between clusters

We have discussed the component of clustering which is based on the new k-means type subspace clustering algorithm *FW-KMeans*. This algorithm is an extension to the well known k-means algorithm by adding a new step to calculate the feature weights for different clusters in the k-means clustering process. We have shown that the new algorithm is scalable to large text data and capable of handling sparse data and finding clusters from subspaces of features. The experiment results on real world text data have shown that this new algorithm performed better than other two subspace clustering algorithms (*PROCLUS* and *HARP*) and *Bisecting-KMeans*.

To effectively present clustering results to users, we have developed a new approach to extracting a few keywords to present the semantics of clusters. The results have shown that it was effective in extracting meaningful keywords.

Our future work is to refine the current text clustering system by including more functions, such as using visual methods to determine the number of clusters and a better way to set up initial cluster centers. We believe that ontology will provide premising solutions on this point. We also plan to test this system on medical literature [35].

REFERENCES

[1] "Textminer." [Online]. Available: http://www.sas.com/technologies/analyticsdatamining/textminer/
[2] "Textanalyst." [Online]. Available: http://www.megaputer.com/products/ta/index.php3
[3] "gcluto." [Online]. Available: http://www-users.cs.umn.edu/ karypis/cluto/gcluto/
[4] "Refviz." [Online]. Available: http://www.refviz.com
[5] W. Fan, L. Wallace, S. Rich, and Z. Zhang, "Tapping into the power of text mining," *the Communications of ACM*, 2005.
[6] B. Ricardo and R. PBerthier, *Modern information retrieval*, 1999.
[7] A. Hotho, A. Maedche, and S. Staab, "Ontology-based text document clustering," Seattle, USA, 2001.
[8] L. Jing, M. Ng, J. Xu, and Z. Huang, "Subspace clustering of text documents with feature weighting k-means algorithm," 2005, pp. 802–812.
[9] C. Fellbaum, "Wordnet: an electronic lexical databases," *The MIT Press*, 1998.
[10] C. Aggarwal, C. Procopiuc, J. Wolf, P. Yu, and J. Park, "Fast algorithms for projected clustering," *Proc. ACM SIGMOD*, pp. 61–72, 1999.
[11] K. Y. Yip, D. W. Cheung, and M. K. Ng, "A practical projected clustering algorithm," *IEEE Transactions on knowledge and data engineering*, vol. 16, no. 11, pp. 1387–1397, 2004.
[12] "20-newsgroups." [Online]. Available: http://kdd.ics.uci.edu/databases/20newsgroups /20newsgroups.html
[13] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering : A review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, 1999.
[14] J. Han, M. Kamber, and A. Tung, "Spatial clustering methods in data mining: a survey," *Geographic Data Mining and Knowledge Discovery*, 2001.
[15] L. Parsons, E. Haque, and H. Liu, "Subspace clustering for high dimensional data: a review," *SIGKDD Explorations*, vol. 6, no. 1, pp. 90–105, 2004.
[16] J. MacQueen, "Some methods for classification and analysis of multivariate observations," 1967, pp. 281–297.
[17] S. Dhillon, J. Fan, and Y. Guan, "Efficient clustering of very large document collections," *Data mining for scientific and engineering applications*, pp. 357–381, 2001.
[18] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of document clustering techniques," 2000.
[19] O. Duda, E. Hart, and G. Stork, *Pattern classification*, 2000.
[20] Y. Zhao and G. Karypis, "Comparison of agglomerative and partitional document clustering algorithms," *Technical report ♯02-014, University of Minnesota*, 2002.
[21] C. Aggarwal and P. Yu, "Finding generalized projected clusters in high dimensional spaces," *Proc. ACM SIGMOD*, pp. 70–81, 2000.
[22] H. Frigui and O. Nasraoui, "Unsupervised lerning of prototypes and attribute weights," *Pattern recognition*, vol. 37, no. 3, pp. 567–581, 2004.
[23] Y. Chan, K. Ching, K. Ng, and Z. Huang, "An optimization algorithm for clustering using weighted dissimilarity measures," *Pattern recognition*, vol. 37, no. 5, pp. 943–952, 2004.
[24] B. M.W., S. Dumais, and G. O'Brien, "Using linear algebra for intelligent information retrieval," *SIAM Review*, vol. 37, pp. 573–595, 1995.
[25] I. Herman, G. Melancon, and M. Marshall, "Graph visualization and navigation in information visualization: a survey," *IEEE Transactions on Visualization and Computer Graphics*, vol. 6, no. 1, pp. 24–43, 2000.
[26] S. Dhillon and S. Modha, "Concept decompositions for large sparse text data using clustering," *Machine learning*, vol. 42, no. 1, pp. 143–175, 2001.
[27] R. Bekkerman, R. El-Yaniv, N. Tishby, and Y. Winter, "Distributional word clusters vs. words for text categorization," *Journal of Machine Learning Research*, vol. 3, no. 7-8, pp. 1183–1208, 2003.
[28] A. McCallum, "Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering," 1996. [Online]. Available: http://www.cs.cmu.edu/mccallum/bow

[29] J. Bezdek, "A convergence theorem for the fuzzy isodata clustering algorithms," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 2, pp. 1–8, 1980.
[30] P. Hague and P. Harris, *Sampling and statistics*, London, 1993.
[31] S. Selim and M. Ismail, "*k*-means type algorithms: a generalized convergence theorem and characterization of local optimality," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 6, no. 1, pp. 81–87, 1984.
[32] W. Li, "Random texts exhibit zipf's-law-like word frequency distribution," *IEEE Transactions on Information Theory*, vol. 38, no. 6, pp. 1842–1845, 1992.
[33] Z. Shi and J. Ghosh, "A comparative study of generative models for document clustering," San Francisco, CA, May, 2003.
[34] I. Katsavounidis, C. Kuo, and Z. Zhang, "A new initialization technique for generalized lioyd iteration," *IEEE signal proceeding, Letters 1(10)*, pp. 144–146, 1994.
[35] D. Swanson, "Medical literature as a potential source of new knowledge," *Bull Med Libr Assoc.*, vol. 78, no. 1, pp. 29–37, 1990.