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

Fast Document Clustering Based on Weighted Comparative Advantage

Jie Ji
Intelligent System Lab
The University of Aizu
Aizuwakamatsu, Fukushima, Japan
d8102102@u-aizu.ac.jp

Tony Y. T. Chan
School of Computing
The University of Akureyri
Iceland
tonychanyt@gmail.com

Qiangfu Zhao
Intelligent System Lab
The University of Aizu
Aizuwakamatsu, Fukushima, Japan
qf-zhao@u-aizu.ac.jp

***Abstract*—Document clustering is the process of partitioning a set of unlabeled documents into clusters such that documents within each cluster share some common concepts. To help with this analysis, concepts are conveniently represented using some key terms. For clustering algorithm, the most costly CPU time has to do with the classification phase. Using words as features, text data are represented in a very high dimensional vector space. We have studied a comparative advantage based algorithm for clustering sparse data in this space, it used one "ruler" instead of k centers to identify the comparative advantage of each cluster and define the cluster label for each document. However, this algorithm only considered the relative strength between clusters, the relationship between terms was ignored. In this paper, we proposed a weighted comparative advantage based clustering algorithm. The experimental results based on SMART system databases show that the new algorithm is better than simple comparative advantage algorithm, without any extra computation time. Compare with *k*-means, not only can it get comparable results but it can also significantly accelerate the clustering procedure.**

***Index Terms*—Document clustering, dimension reduction, key term extraction, sparsity, k-means, weighted comparative advantage.**

I. Introduction

Document clustering is the process to partition a set of unlabeled documents into some categories or clusters. To analyze the documents based on the clustering results, it is expected that all documents in a cluster have some shared concepts. This shared concept is often represented as the centroid. *K*-means is a well known algorithm for unsupervised clustering [1]-[3]. In *k*-means, clusters are obtained based on the mean square error (*MSE*) criterion. These clusters are good in the sense that all documents in each cluster are similar to each other. However, since the similarity is often defined based on all terms, the sparsity fixes there are a lot of inessential parts in each centroid. To cluster these documents, we believe that only small set of representative key terms should be concerned [4]. We have studied a comparative advantage based algorithm for clustering sparse data, it uses one "ruler" instead of k centers to identify the comparative advantage of each cluster and define the cluster label for each document. However, that algorithm only considers the relative strength between clusters, the relationship between terms is ignored. In this paper, we proposed a method which uses weighted "ruler" instead of simple "ruler". To verify the effectiveness of the proposed method, we conducted several experiments on three SMART databases. The experimental results show that the proposed method can generate better results than simple comparative advantage.

II. Vector Space Model for Text Data

In this study, we use term frequency - inverse document frequency (tf-idf) model to represent each document as a vector [5]. To transform document into vector, the first step is morphological analysis [6]. The purpose of this step is to segment the smallest units of syntax from the whole text. Since in most languages, words are accepted as the smallest units of syntax, the output of morphological analysis is usually a bag-of-words for representing the given document. Of course, not all words are useful for document clustering. We eliminate the "stopwords" usch as "a", "does", "ok", etc [7].

After morphological analysis, term frequency and document frequency are counted for each word. Here, term frequency tf_{ji} is the frequency of the ith term in the jth document and document frequency df_i is the number of documents that contain the ith term. Since the sparsity of text data, a lot of words only appear in very small portion of documents, we usually eliminate the words whose df is less than a threshold r_0, e.g., 0.2%.

Suppose m unique words remain after above elimination. Then document j is represented as a vector $doc_j = (v_1, v_2, \ldots, v_m)^t$ in the term-space R^m, where m is the total number of terms used in the whole document set, and

$$v_i = tf_{ji} \times g_i \times s_j,$$

where g_i is global weighting component depends on df_i and s_j is the normalization component for v_i. In tf-idf model, $g_i = \log(n/df_i)$. To represent long documents and short documents equally, each document vector is normalized to have unit L^2 norm, that is,

$$s_j = \left(\sum_{j=1}^{d} (tf_{ji} g_i) \right)^{-1/2}$$

After finding the feature vectors, vector based clustering algorithm can be used for clustering. The most popular algorithm for this purpose is k-means algorithm [3]. The objective

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

of k-means is to minimize the total intra-cluster variance, or the mean squared error (*MSE*) function defined as follows:

$$MSE = \frac{1}{k}\sum_{i=1}^{k}\sum_{\forall x_j \in S_i}\left(x_j - \mu_i\right)^2 \quad (1)$$

where k is the number of clusters, S_i is the ith cluster, and μ_i is the centroid or mean point of S_i.

The most common form of k-means uses an iterative refinement heuristic known as Lloyd's algorithm [8]. Lloyd's algorithm starts by partitioning the data points into k initial clusters, either at random or using some heuristic. It then performs the following two steps iteratively:

- Step 1: Calculate the centroid of each cluster.
- Step 2: Construct a new partition by associating each data point with the closest centroid.

Usually, the above iteration is terminated if the data points no longer switch between clusters, or alternatively, if the centroids no longer change.

Note that in Step 2, we need a distance measure to find the closest centroid for the given data point. Usually, Euclidean distance is used. However, for high dimensional document vector clustering, cosine distance is better [9]. Since the feature vectors have a;ready been normalized, we can just use the inner product to measure the similarity between two vectors:

$$\cos(d_i, d_j) = \frac{< d_i, d_j >}{\|d_i\| \times \|d_j\|} =< d_i, d_j > \quad (2)$$

If two data points are very similar, the angle between them is close to zero, and the inner product will be close to one. Hence, the inner product $< d_i, d_j >$ is oftern known as the "cosine similarity", it is widely used in text mining and information retrieval [7][10].

III. Comparative Advantage

In this section, we first review the simple comparative advantage (CAS) based algorithm. Compared with *k*-means, the algorithm encouraged each cluster to have some dominant representative key term, the terms with only small effects are tend to be ignored. Then weighted CA based algorithm (CAW) is introduced. Compare with CAS, CAW considers the relationship between terms, in this sense, the clustering results are expected to be more meaningful.

A. Term Distribution

Comparative advantage is based on the concept of term distribution. Basically, the term distribution for the ith cluster S_i is an m-dimensional vector with the jth element being the frequency of the jth term in this cluster. Remember that m is the dimensionality of vector space R^m.

To make the concept easier to understand, Table I shows 4 document vectors. We want to partition these 4 documents into 2 clusters. For example, if the first two documents are assigned to S_1, and the last two are assigned to S_2, sum up the term frequency for all terms in each cluster, we get the term distributions as shown in Table II. If we put doc1 and doc3 into S_1, and doc2 and doc4 into S_2, we can get another term distributions as shown in Table III.

TABLE I
Term Matrix for the Example

	birth	liver	life	steam	shock	speed
$doc1$	1	1				
$doc2$	1		1			
$doc3$				1		1
$doc4$				1	1	1

TABLE II
Term Distributions of the Two Cluster in the Example

	Distribution1	Distribution2
birth	2	0
liver	1	0
life	1	0
steam	0	2
shock	0	1
speed	0	2

TABLE III
Term Distributions Corresponding to Another Partition

	Distribution1	Distribution2
birth	1	1
liver	1	0
life	0	1
steam	1	1
shock	0	1
speed	1	1

The term distributions can be obtained straightforwardly once a partition is given. We represent a partition using a number string of length n, where n is the number of documents. Each number in the string takes value from $[1, k]$. If the jth number of string is i, which means the jth document belongs to the i-th cluster.

For the above example, the first partition can be represented as [1 1 2 2]. That is, the first two documents are assigned to the first cluster, and the last two are assigned to the second cluster. The second partition is [1 2 1 2], and the meaning can be interpreted straightforwardly.

Our goal is to find a partition so that each cluster uses a different set of key terms. In this sense, the partition given in Table II is better than the one given in Table III. Since in Table II, "birth", "liver" and "life" are in same cluster indicating this cluster is mainly taking about medical technology, and the second cluster contains words "steam","shock" and "speed", which indicates this is a aeronautical system cluster.

Figure 1 shows graphically result of term distributions of CISI, MEDLINE and CRANFILED of database CLASSIC3. We will give a description of this data collection in Section IV. Figure 2 show terms distributions by randomly assigned the cluster label for each document. Intuitively speaking, the term

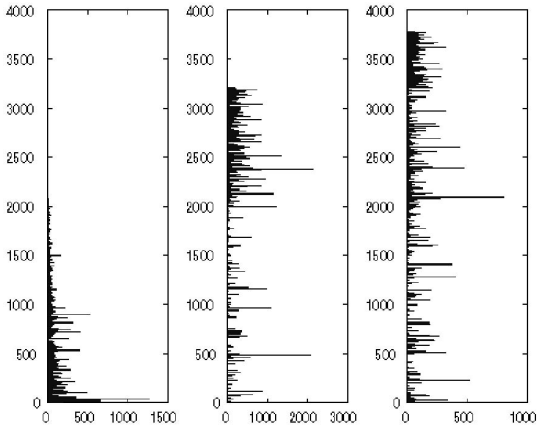

Fig. 1. Term distributions initialized according to teacher signal

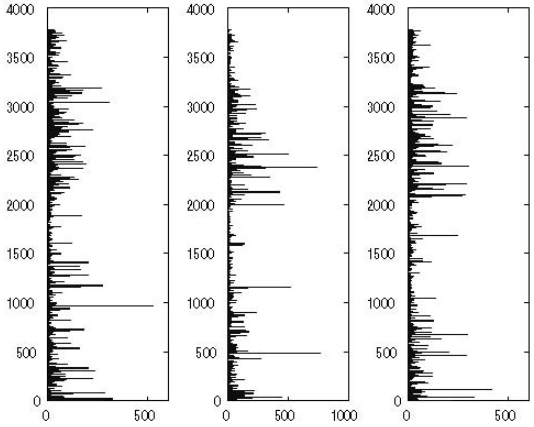

Fig. 2. Term distributions initialized randomly

distributions initialized by using class label is much better than the second one. Because from the second term distributions we can not find out useful representative key words.

B. Simple Comparative Advantage

From Figure 1 we can find out that different classes featured different key terms. The sum of these key terms' weights are bigger than other classes. To get a good partition, we should find and keep this kind of "advantages" or "strength". Intuitively speaking, if the weight of term i of term distribution j is larger than others, term i is the advantage of term distribution j.

However, bigger weight value doesn't mean it is an advantage absolutely. Let's take an example [11]. Suppose that there are two men living in a small island. They have to do something to feed themselves. There are two possible ways for them to survive, hunting or planting. The first man is young and strong, and he can get \$60 from hunting and \$20 from planting one day. The second man is old and weak, and he can get only \$20 from hunting and \$10 from planting one day. If one man is hunting, then another man cannot hunt at the same time. And one person can only do one thing at the same time. The problem is how to maximize the total profit?

TABLE IV
ATTRIBUTES OF THE TWO MEN

	Hunting		Planting	
	AA	CA	AA	CA
Young	\$60	3/4	\$20	1/4
old	\$20	2/3	\$10	1/3

From Table IV we can see, the young man can earn more money both on hunting and planting (\$60 per day and \$20 per day). However, if these values are divided by the summation of his ability (\$80 per day), we can get two ratios: $3/4$ and $1/4$. We can get two ratios of the old man using same way: $2/3$ and $1/3$. Which means, although the young man can earn more money on planting than the old man, compare to themselves total ability, the old man is better at planting. Actually, to let the young man hunt and let the old man plant, they can maximize the profit, totally \$70 per day.

In economics, this kind of modified advantage is called *comparative advantage* [12]. If we take term distribution as a human's ability, the basic idea of comparative advantage is to scale the term distribution by the total sum of the corresponding cluster, so that the clusters with different sizes can compete equally. Specifically, the term distribution of each cluster is normalized as follows:

$$W'_{ip} = \frac{W_{ip}}{\sum_{q=1}^{m} W_{iq}}, \quad i = 1, 2, \ldots, k; p = 1, 2, \ldots, m \tag{3}$$

If W'_{ip} is the maximum among all $i = 1, 2, \ldots, k$, we say that the ith cluster is comparatively stronger than other clusters in the pth term.

We use a "ruler" to indicate the comparative advantages for term distributions. The pth element of the ruler is defined as follows:

$$ruler[p] = \arg \max_{1 \le j \le k} W'_{jp}, \quad p = 1, 2, \ldots, m \tag{4}$$

That is, if the ith cluster is comparatively the strongest in the pth term, the pth element $ruler[p]$ of the ruler will be i.

For any given document represented by an m-dimensional feature vector $doc = (v_1, v_2, \ldots, v_m)$, its label is defined as follows:

$$label[d] = \arg \max_{1 \le j \le k} V_i \tag{5}$$

where V_i is given by

$$V_i = \sum_{\substack{1 \le q \le m \\ ruler[q] = i}} v_q \tag{6}$$

C. Weighted Comparative Advantage

The CA we discussed above is called simple comparative advantage (CAS). It is because for each term, the CAS only indicates the "strongest" cluster, but without the information of "how strong". Actually for equation 3, W'_{ip} also includes the comparative strength between terms. If W'_{ip} is the maximum among all $p = 1, 2, \ldots, m$, the pth term is the most important term for representing the ith cluster. Without this kind of information, the result will lose some accuracy. Let's take an example.

Suppose we have two clusters, the ith term's weight of cluster 1 $W'_{1i} = 0.08$, for cluster 2 $W'_{2i} = 0.09$; and for term j, $W'_{1j} = 0.5$ and $W'_{2j} = 0.1$. For simple ruler, we can get $ruler[i] = 2$ and $ruler[j] = 1$. If a document vector has same value both on v_i and v_j, according to equation 6, $V_i = V_j$, which will be difficult to define label for this document.

To solve this problem, we can use *weighted comparative advantage* (CAW) strategy. The ruler using this strategy is called weighted ruler, it is a $2 \times m$ dimensional matrix, the first row is as same as simple ruler:

$$ruler_w[0][p] = ruler_s[p] \tag{7}$$

and

$$ruler_w[1][p] = W'_{ip}, \quad i = ruler_w[1][p] \tag{8}$$

where $suler_s$ is simple ruler and $ruler_w$ is weighted ruler. Then the V_i is defined by:

$$V_i = \sum_{\substack{1 \le p \le m \\ ruler_w[0][q] = i}} v_q \times ruler_w[1][p] \tag{9}$$

if we use weighted ruler.

From equation 6 and 9 we can see, to define cluster label for a given document, both of them only need to calculate m times.

D. Clustering method

Based on the above discussions, we proposed a new clustering algorithm similar to k-means. The algorithm also uses Lloyd's algorithm, and is described as follows:

Step 1: Initialize the clusters.

Step 2: Calculate the ruler from the current partition.

Step 3: Use the ruler to get a new partition.

The program will stop if the ruler does not change, otherwise return to step 2. Note that in Step 1, we may initialize the clusters at random, or select document randomly as the initial term distribution, or using some existing clustering algorithm, say k-means. Intuitively speaking, after clustering, all documents in the same cluster will share the same important key terms.

TABLE V
PROPERTIES OF DATA SETS

	Number of document	Truncation	Dimension
CLASSIC3	3983	$df/n > 0.2\%$	3780
Original CLASSIC3	3893	all	19929
MEDLINE	1033	all	7256
CISI	1460	all	11012
CRANFILED	1400	all	5040
NSF3	4303	$df/n > 0.2\%$	5392
Original NSF3	4303	all	18721
ASTRONOMY	846	all	5736
BIOLOGY	1954	all	12548
COMPUTER	1503	all	8489

TABLE VI
CONFUSION MATRIX OF 3 CLUSTERS ON CLASSIC3

	CISI	CRANFILED	MEDLINE
$\pi_1^\dagger$	1448	13	21
$\pi_2^\dagger$	8	1386	6
$\pi_3^\dagger$	4	1	1006
Total	1460	1400	1033

IV. EXPERIMENTAL RESULTS

A. General considerations

To verify the proposed method, we obtained two subsets named as CLASSIC3 and NSF3. CLASSIC3 is a subset from **SMART** system - one of the most popular test beds where the vector space based algorithm is successfully implements. It contains 3893 documents by merging the MEDLINE, CISI and CRANFILED sets. MEDLINE contains 1033 abstracts from medical journals, CISI constains 1460 abstracts from information retrieval papers, and CRANFIELD contains 1400 abstracts from aeronautical systems papers (ftp://ftp.cs.cornell.edu/pub/smart). NSF3 contains 4303 abstracts of the grants awarded by the Nation Science Foundation. We collected 846 abstracts from astronomy area, 1954 abstracts from biology area and 1503 abstracts from computer science area.

We removed the title and author, kept abstract information. Then the documents were changed into vector as in Section 2. After removing common English stopwords, and merging mutual words, the CLASSIC3 collection contained 19929 unique words and the NSF3 collection contained 18721 unique words. Then we use a naive truncation method to reduce the dimensionality: To remove the words whose document frequency is less than 8 (roughly 0.2% of the documents). Table V shows the properties of original data set and dimensional reduced data set.

We clustered data from 3 to 16 clusters, totally 14 cases, and use criteria functions to evaluate the results, respectively.

For each cluster case, we conducted 100 times runs and calculated the average value. Since both of the proposed algorithm and *k*-means algorithm are using Lloyd's searching algorithm, they are initialization sensitive algorithm. In order to keep consistency, for each run, they used the same randomly

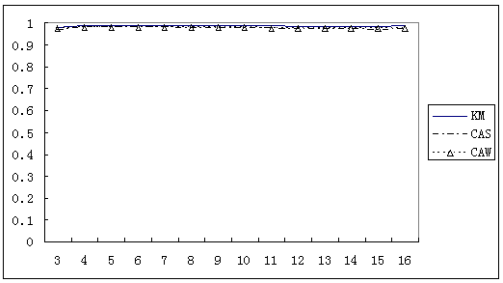

Fig. 3. Precision of CLASSIC3

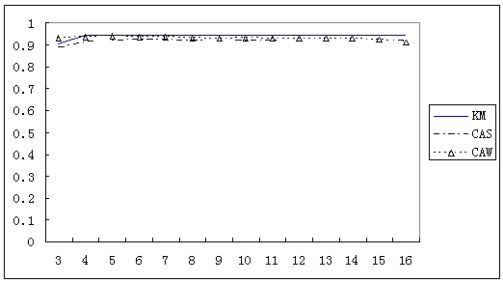

Fig. 4. Precision of NSF3

initialized point. That is, to select k documents as the mean vector for the first step of k-means, and for term distributions of the proposed method also.

The precision, overlap rate and CPU times per iteration are used for criteria functions in this study.

For each partition, we can get a $k \times 3$ dimensional confusion matrix with the element a_{ij} represents the number of documents in the ith cluster with class label j. Each cluster is assigned a class label by finding the major class in this cluster. The precision can be calculated from confusion matrix by using the following equation:

$$p = \frac{\sum_{i=1}^{k} \max(a_{ij})}{n}, \quad (j = 1, 2, 3) \tag{10}$$

The overlap rate is a simple way to evaluate the quality of partition. It is defined as:

$$OR = \frac{1}{M} \Big| \bigcap_{i=1}^{k} \Omega_i \Big| \tag{11}$$

where Ω_i is the set of representative key terms for the ith cluster, and M is the number of representative terms used in each cluster, $M < m$. For our experiments, the top 38 important terms were chosen. If OR equals to 1, all clusters will share the same key terms. If OR equals to 0, the sets of important key terms in all clusters are completely different.

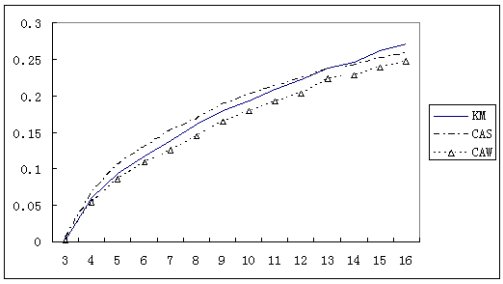

Fig. 5. Overlap Rate of CLASSIC3

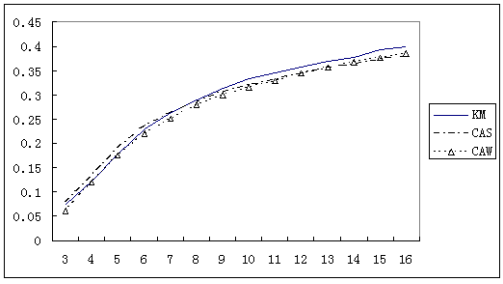

Fig. 6. Overlap Rate of NSF3

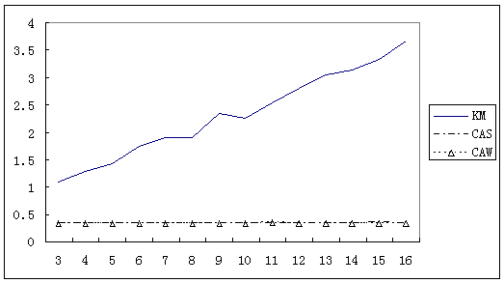

Fig. 7. CPU time of CLASSIC3

B. Experimental results

The confusion matrices in Table VI shows that the 3 clusters $\pi_1^\dagger$, $\pi_2^\dagger$ and $\pi_3^\dagger$ produced by CAW clustering algorithm. We can see from the table that the proposed method do clustered the given document set in a reasonable way.

The experimental results are given in Figure 3 to 8. From these results we can make the following observations:

The proposed methods and k-means algorithm are comparable in sense of precision. However, for bigger number of clusters, our method may not work well. It is because the proposed method only uses an m dimensional "ruler" to classify documents to k clusters. For each cluster, there are m/k dimensions on average. If k is small, major differences are enough to distinguish the class label of document, if k is big, the information for identifying a cluster will be smaller. Let's take an extreme case for example, if $k = m, m = n$, the m dimensional clustering problem degrades to a "1" dimensional clustering problem. That is to say, for each cluster, only one dimension is used to calculate distance. This kind of dimensionality reduction will lose some information.

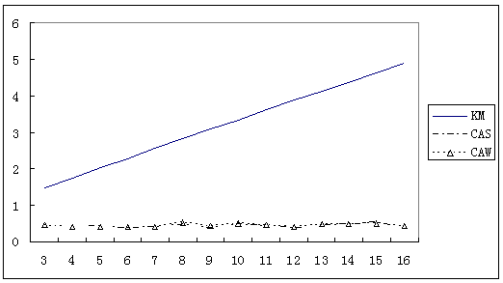

Fig. 8. CPU time of NSF3

Compare with the high dimensionality of term-space, the number of cluster k is a small enough number. For each cluster, enough information will be assigned for each cluster. The proposed methods made the best use of the sparsity of text data.

The CAW can get partitions with higher precision than CAS. The reason has been described in III.C. Intuitively speaking, comparative advantage indicates the relative strength between clusters, and the weight of comparative advantage indicates the strength between weight of terms. Moreover, from Figure 5 and Figure 6 we can see it can also get a smaller overlap rate than both CAS and *k*-means.

Since all the methods use Lloyd's algorithm, the average iteration times of proposed algorithms are similar as *k*-means. However, the CPU time for each iteration is very different. Figure 7 and Figure 8 show the results of CPU time *for each iteration* in seconds. As we know, for most clustering algorithms, the major computation involves comparison of two vectors. Suppose there are n documents in a R^m dimensional term-space, and the number of cluster is k, for classical *k*-means, we may get the following C code implements the classical *k*-means algorithm with double-loop:

Classification function of K-means

```
for (i=0;i<k;i++){
    for (j=0;j<m;j++){
        distance[i] +=
        doc[j]*center[i][j];
    }
}
```

From the code above we can find that for *k*-means, the computation complexity of one iteration is $O(k \times m)$.

The classification function by a ruler uses a single loop:

Classification function of CAS

```
for (j=0;j<m;j++){
    distance[ruler[j]] +=
    doc[j]*ruler[j];
}
```

If we use weighted ruler, actually the procedure won't cost any extra time, the C code is:

Classification function of CAW

```
for (j=0;j<m;j++){
    distance[ruler[0][j]] +=
    doc[j]*ruler[1][j];
}
```

In comparative advantage based algorithm, the computation cost for one iteration is $O(m \times n)$, since one ruler represents all k centers. Theoretically speaking, the proposed method can reduce k times in classification phase.

The experimental results on the same PC shows that the computation cost of CAS and CAW are almost same. For 3 clusters case, the proposed methods accelerated about 1.8 times than *k*-means; and about 12.9 times faster than *k*-means for 16 clusters case. Note that the CPU time is not just classification time, it also contains other procedures.

V. Conclusion

In this paper, we have proposed a fast clustering algorithm based on weighted comparative advantage (CAW) for sparse text data. Compared with *k*-means, it can reduce the computation cost by compressing k centroids into one simple "ruler". The results show that the proposed method can get a comparable precision with *k*-means but much faster. Compare with simple comparative advantage based clustering algorithm (CAS), CAW can not only get a partition with higher precision, but also a smaller overlap rate in sense of representative key terms. Moreover, it doesn't require any extra computation time.

As a future study, we would like to propose more efficient and effective search algorithms to get better clustering results. The proposed algorithm will be applied for dimension reduction for text data. And the performance will be compared with SVD, PCA or some other dimension reduction methods.

References

[1] J. B. MacQueen, "Some Methods for classification and Analysis of Multivariate Observations", Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, pp.281-297, 1967.

[2] J. A. Hartigan, Clustering Algorithms. Wiley, 1975.

[3] J. A. Hartigan and M. A. Wong, "A *K*-Means Clustering Algorithm", Applied Statistics, Vol.28, No.1, pp.100-108, 1979.

[4] Jie Ji, Qiangfu Zhao, "Comparative Advantage Approach for Sparse Text Data Clustering", Proceeding of IEEE 9-th International Conference on Computer and Information Technology, Xiamen, China, 2009.

[5] Salton, G. and Buckley, C, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management* 24(5), pp.513-523, 1988.

[6] Bauer, Laurie, Introducing linguistic morphology, 2nd Ed., Washington, D.C., Georgetown University Press, 2003.

[7] Frakes, W.B. and R. Baeza-Yates, Information Retrieval: Data Structures and Algorithms, Prentice Hall, Englewood Cliffs, New Jersey, 1992.

[8] Stuart P. Lloyd, "Least Squares Quantization in PCM," IEEE *Transactions on Information Theory*, vol.28, no.2, pp.129-137, 1982.

[9] Inderjit S. Dhillon and Dharmendra S. Modha, "Concept Decompositions for Large Sparse Text Data Using Clustering", Machine Learning, vol. 42, Jan. 2001, pp. 143-175, doi: 10.1023/A:1007612920971.

[10] Salton, G. and M. J. McGill, Introduction to Modern Retrieval, McGraw-Hill Book Company, 1983.

[11] A. O'Sullivan & S.M. Sheffrin, Economics. Principles & Tools, 3th Ed., Prentice Hall, 2002.

[12] Hardwick, Khan and Langmead, An Introduction to Modern Economics, 5th Ed., Financial Times/ Prentice Hall, 1999.