Efficient Clustering Approach using Incremental and Hierarchical Clustering Methods

M. Srinivas and C. Krishna Mohan

Abstract−**There are many clustering methods available and each of them may give a different grouping of datasets. It is proven that hybrid clustering algorithms give efficient results over the other algorithms. In this paper, we propose an efficient hybrid clustering algorithm by combining the features of leader's method which is an incremental clustering method and complete linkage algorithm which is a hierarchical clustering procedure. It is most common to find the dissimilarity between two clusters as the distance between their centorids or the distance between two closest (or farthest) data points. However, these measures may not give efficient clustering results in all cases. So, we propose a new similarity measure, known as cohesion to find the intercluster distance. By using this measure of cohesion, a two level clustering algorithm is proposed, which runs in linear time to the size of input data set. We demonstrate the effectiveness of the clustering procedure by using the leader's algorithm and cohesion similarity measure. The proposed method works in two steps: In the first step, the features of incremental and hierarchical clustering methods are combined to partition the input data set into several smaller subclusters. In the second step, subclusters are merged continuously based on cohesion similarity measure. We demonstrate the effectiveness of this framework for the web mining applications.**

I. INTRODUCTION

One drawback of the partitional clustering is the difficulty in determining the optimal number of clusters(k). Incremental clustering is an efficient method and runs in linear time to the size of input data set. In most related studies, the dissimilarity between two clusters is defined as the distance between their centroids or the distance between two closest data points. Hierarchical clustering algorithms create a hierarchical decomposition of data set based on some criterion. The decomposition can be described as a dendrogram that represents a series of nested partitions. A partition is obtained by cutting the dendrogram at some desired level. Hierarchical algorithms do not suffer from the problem of choosing a pre-specified number for the output clusters [3].

M.Srinivas with the Indian Institute of Technology, Hyderabad-502025, India (phone: +91 9989449091; email: sreeconf@gmail.com).
C. Krishna Mohan with the Indian Institute of Technology, Hyderabad-502205, India (email: ckm@iith.ac.in).

Hierarchical clustering algorithms can differ in their operation. Agglomerative clustering methods start with each object in a distinct cluster and successively merge them to larger clusters until a stopping criterion is satisfied. Alternatively, divisive algorithms begin with all objects in a single cluster and perform splitting until a stopping criterion is met. Both agglomerative and divisive hierarchical algorithms are static in the sense they never undo what was done previously, which means that objects which are committed to a cluster in the early stages, cannot move to another cluster. In other words, once a cluster is split or two clusters are merged, the split objects will never come together in one cluster or the merged objects will be never in the same cluster, no matter whether the splitting or the merging is the correct action or not. But in practice, some splitting or merging actions may not be correct and there is a need to rearrange the partition. This problem is a cause for inaccuracy in clustering, especially for poorly separated data sets.

In this paper, we present a hybrid clustering algorithm namely, Leaders complete linkage algorithm (LCL) that combines the advantages of hierarchical clustering and incremental clustering techniques. Instead of rearranging all objects like what partitional algorithms usually do, in each iteration of the clustering, some objects but all are moved from one cluster to another by the way of splitting a cluster or merging two clusters. It can start with a single cluster containing all objects or start with each object in a distinct cluster. At each step during the clustering, the quality of the current partition is examined as well as the quality of the partition after splitting one cluster or merging two clusters. If the quality of the partition is improved after the splitting or the merging, then a further splitting or merging will be performed. Otherwise, the clustering will terminate and the current partition is the final clustering result. The idea behind the mix of splitting and merging is to allow amendment to the previous clustering result so that high quality clustering can be achieved. The amount of information available on the Web has been increasing dramatically in recent years. At present the main problem that the Web users face with is no longer the lack of information, but the way of finding information that can meet their specific needs. Recently, many researchers have

978-1-4244-8126-2/10/$26.00 ©2010 IEEE

focused their study on applying data mining techniques to Web server logs for automatically generating user navigation patterns such as popular navigation sessions, item association rules, page clusters, and user clusters [1 -7], [9]. A variety of knowledge discovery techniques have been applied to obtain navigation patterns. Clustering is one of the popularly used techniques for grouping user navigation sessions or Web pages. The LCL algorithm proposed here has been experimentally tested on a set of real Web server log data to find Web page clusters. Hybrid clustering methods are used to overcome the misclassification problem with large data sets. The basic technique is to first find suitable prototypes from the large data set and then apply the clustering method using only the prototypes.

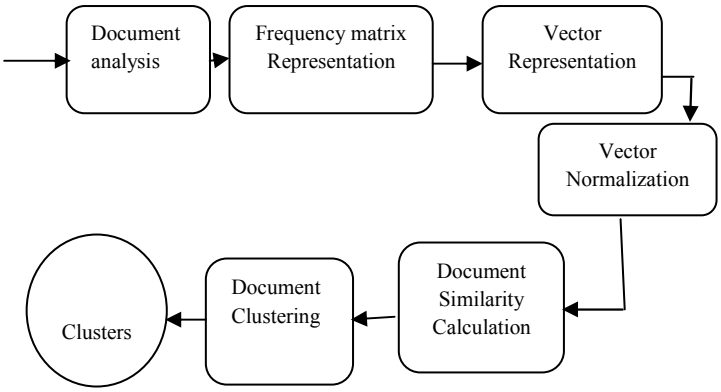

Fig.1. Overall System Design

Fig. 1 shows the overall system design of text document clustering process. In this process, documents are taken as the input. These input documents are analyzed or preprocessed by the document analyzer. In this analysis phase, features of the each document are selected and represented in terms of frequency matrix. Further, this frequency matrix is converted into vector representation and the vectors are normalized. Then a similarity measure is applied on each document and a similarity matrix is constructed. Finally, the text documents can be clustered based on a similarity measure.

The rest of the paper is organized as follows. Section 2 introduces the hybrid clustering algorithm. Section 3 presents the experimental results of applying the hybrid clustering algorithm to Web page clustering. Comparisons to related work are also provided in this section. Finally, Section 4 summarizes the paper.

II. HYBRID CLUSTERING ALGORITHM

A. Leader Clustering Algorithm

Leader clustering algorithm [2] is a single pass algorithm and very efficient in terms of computational requirements. When leader algorithm is applied on the data set, it chooses one data object and assumes it as a first leader. Then, it chooses all the data objects one by one and compares it with the leader. The comparison is based on the Euclidean distance between leader and the current data object. Whether to put the current data object within the cluster lead by that particular leader or make that data object as a new leader is decided based on the threshold value. This process will continue until the entire data set has been processed. Following is the leader algorithm:

Algorithm: Leader Clustering
Input: Data set, Threshold value (*Th*).
Output: Number of Clusters (*k*).
Initialize a leader, add it to the leader list and
set leader counter L =1
Do for all patterns i =2 to N
{
Calculate the distance between pattern i and all leaders
Find the nearest leader
If (distance between pattern i and nearest leader<Threshold) then
{
Assign pattern i to the nearest leader
Label the cluster number
Add pattern i to member list of this cluster
Increment member count of this cluster
}
else
{
Add it to the leader list
Increment leader counter $L = L + 1$
}
}

For a given threshold value (*Th*), leaders method [2] works as follows. It maintains a set of leaders L, which is incrementally built. For each pattern x in dataset (D) if there is a leader $l \in L$ such that distance between x and l is less than *Th*, then x is assigned to the cluster represented by l. Note that even if there are many such leaders, only one (the first encountered one) is choosen. Because of this the cluster represented by a leader is of semi-spherical shape. If there is no such leader then x itself becomes a new leader and is added to L. Along with each leader, a count indicating number of patterns that are grouped under the leader is also stored.

Leader clustering algorithm scans the input data set only once and gives the result very quickly. The advantage with leaders clustering algorithm is that there is no need to

know the number of clusters in advance. Leaders clustering algorithm finds appropriate number of clusters based on the threshold value (*Th*). The leader algorithm requires only $O(n)$ time to get the clusters from the data set with n number of data objects. The drawbacks with this algorithm are: (1) Threshold value must be very appropriate, (2) output depends on the order in which data is presented, and (3) arbitrary shape of the cluster may not be possible to determine. Leader's algorithm produces only the circular clusters. Also, the result of leader clustering algorithm is highly affected by the presence of noise.

B. Hierarchical Clustering Algorithms

As its name implies, a hierarchical clustering algorithm establishes a hierarchical structure as the clustering result. Owing to their good quality of clustering results, hierarchical algorithms are widely used especially in document clustering and classification. The outline of a general hierarchical clustering algorithm is given below:

Hierarchical Clustering Algorithm:

1. Initially, each data point forms a cluster by itself.
2. Repetitively merge the two closest clusters.
3. Output the hierarchical structure that is constructed.

Most existing hierarchical clustering algorithms are variations of the single-link and complete-link algorithms. Both algorithms require time complexity of $O(n^2 \log n)$. Where n is the size of the input data set. These algorithms differ in the way they characterize the similarity between a pair of clusters. A single-link clustering algorithm differs from a complete-link clustering algorithm in the intercluster distance measure. The single-link algorithm uses the distance between the two closest points of the two clusters as the intercluster distance, i.e,

$$dist\ (Ci,Cj)=min\{dist(oi,oj)|\ oi \epsilon Ci,\ oj \epsilon Cj)\}, \qquad (1)$$

where as the complete-link algorithm uses the distance of two farthest points as the intercluster distance, i.e,

$$dist\ (Ci,\ Cj) =max\{dist(oi,\ oj)|\ oi \in Ci,\ oj \in Cj)\}. \qquad (2)$$

In either case, two clusters are merged to form a larger cluster based on minimum distance criteria. The complete-link algorithm produces tightly bound or compact clusters. The single-link algorithm, by contrast, suffers from a chaining effect. It has a tendency to produce clusters that are elongated. The clusters obtained by the complete-link algorithm are more compact than those obtained by the single-link algorithm. From a pragmatic viewpoint, it has been observed that the complete-link algorithm produces more useful hierarchies in many applications than the single-link algorithm.

C. Proposed Approach

Several clustering methods have been proposed to combine the features of two different clustering algorithms. In general, these algorithms first partition the input data set into m subclusters and then construct a hierarchical structure based on these m subclusters. In this paper we propose a new hybrid clustering algorithm known as leaders complete-linkage (LCL) to merge leaders algorithm and hierarchical clustering algorithm called complete-linkage. In LCL clustering, we first apply Leaders algorithm to produce sub-clusters represented by leaders. Next complete-linkage algorithm is applied on each leader to produce final arbitrary shaped clusters.

(i). Similarity Measure between Subclusters

In this paper, we propose a new measure of similarity between two subclusters known as cohesion which is intrinsically different from other prior measures. Cohesion [8] is more appropriate for an intercluster similarity measure because it does not judge the similarity of two subclusters by considering only few data points. Rather, the cohesion measure takes the distributions of the two clusters into account in order to find the similarity.

As shown in Fig. 2, the distance between the centroids of the two clusters in Fig. 2a and that of the two clusters in Fig. 2b are the same. The two clusters shown in Fig. 2b are more inclined to be merged together. So, we propose a new similarity measure namely, cohesion based on the joinability of two clusters, referring to the existence of a data point. Conceptually, joinability is the merging inclination of two clusters according to the existence of a shared data point. Thus, joinability is expected to have the following properties: (1) Data points located closer to the boundary of the two clusters are more important, and (2) The merging inclination should not be determined by only a few points, i.e., the value of joinability should not vary dramatically.

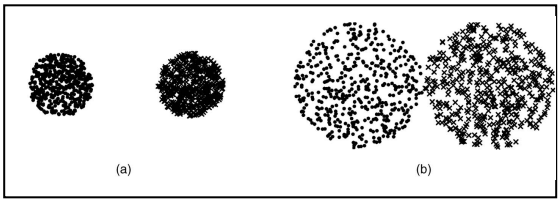

Fig.2. Illustration for subclusters that have different cohesion values. (a) Two clusters with smaller cohesion. (b) Two clusters with greater cohesion.

Formally, we have the following definition for joinability:

Definition 1. The joinability of the two clusters (*Ci* and *Cj*) referring to the existence of the point *p* with location *v* is defined as:

$$Join\ (p,\ Ci,\ Cj) = min\ (f_i(v), f_j(v)), \qquad (3)$$

where f_i and f_j are the probability density functions (pdfs) of the distributions in Cluster *Ci* and *Cj*, respectively. An illustration of joinability is shown in Fig. 3.

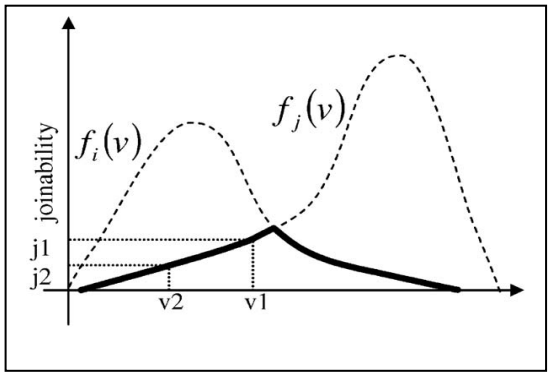

Fig. 3. An illustration for the meaning of joinability

The joinabilities of the data points at *v1* and *v2* are *j1* and *j2*, respectively. With the notion of joinability, the definition of cohesion of two clusters is given below:

Definition 2. The cohesion of two clusters (*Ci* and *Cj*) is defined as

$$chs\ (Ci,\ Cj) = \frac{\sum_{p \in Ci, Cj} join(p, C_i, C_j)}{|C_i| + |C_j|}, \qquad (4)$$

where |*Ci*| is the size of Cluster *Ci.*

In general, the probability density function of $f(v)$ can be evaluated in constant time. Thus, the time complexity of the computation of cohesion of two clusters (*Ci* and *Cj*) is linear to the size of the two clusters, i.e., $O(|Ci| + |Cj|)$.

(ii). Algorithm LCL

Now, we describe the proposed Leaders complete-linkage clustering (LCL) algorithm as follows:

Input: Input data set, size of the data set *n*, number of subclusters *m*, and desired number of clusters *k*.

Output: Hierarchical structure of the *k* clusters.

1. Apply Leaders Algorithm on the input data set to obtain *m* subclusters by using appropriate threshold value.

2. Apply the complete-link clustering algorithm on the *m* subclusters produced in Step 1 with cohesion as the similarity measure and stop when *k* clusters are obtained.

Algorithm LCL is a two-phase clustering algorithm. In the first phase, it adopts the Leader algorithm to divide the input data set into *m* subclusters. At the beginning of phase 2, it obtains the cohesions of these *m* subclusters produced in the first phase. Then, a complete-link clustering algorithm based on cohesion to obtain the final clusters is performed.

In algorithm LCL, the parameter *m*, i.e., the number of subclusters, is the only additional parameter. The desired clustering results can be obtained by adjusting the value of *m*. It is clear that the value of parameter *m* falls in the range of (*k*, *n*). When *m* = *k*, algorithm LCL is degenerated to the leader clustering algorithm. When *m* approaches *n*, this algorithm is reduced to the complete-linkage algorithm. It is known that the leader algorithm is good for obtaining clusters of circular shape, while the complete-link algorithm is able to find clusters of any shape. With prior knowledge, we can make the clustering algorithm adapt to various inputs by adjusting the parameter *m*. The algorithm LCL leads to a better clustering results than those by most prior methods, showing not only the generality but also the advantage of algorithm LCL over the leader and complete-link methods.

(iii). Time and Space Complexities

In step 1, Leaders algorithm takes time $O(n)$ to apply the Leaders algorithm on the input data set to obtain *m* subclusters. Then, at the beginning of step 2, similarity measure between any two subclusters based cohesion is determined. Recall that the time complexity of each evaluation is linear to the size of the clusters. The complete-link clustering algorithm is applied on *m* subclusters and has time in the order of $O(m^2\ log\ m)$. The time complexity of algorithm LCL is therefore $O(n) + O(m^2\ logm) = O(n + m^2\ log\ m)$. In the first step, we only need the memory space to store input data set and group relationship of subclusters, which requires the memory space of $O(n^2)$. In the second step, space is required for the complete-link clustering algorithm. Thus, the space complexity is $O(m^2)$. Since *m* is always smaller than *n*, the total space complexity of algorithm LCL is $O(n^2)$.

III. EXPERIMENTAL RESULTS

In this section, we first describe the data used in the experiments and then the methods to evaluate the performance of the algorithm. Finally, the results obtained in

the experimental studies are discussed. The experiments were conducted using four web server log files, 991001.gz, 991002.gz, 991003.gz, and 991004.gz, which contain the visiting records to a course web site in the department of computer science at the University of JNTUEC: http://www.cs.JNTUEC.edu/educationlcourses/142/96a. The log files were obtained from the Web site: http://www.cs.JNTUEC.edu/ai/adaptive-data2/, archived by the authors of [6].

Each log file contains the visiting records of many users [or visitors] within a day. For each log file, we first preprocess the file, identify the pages and then consider the sessions visited by each user. The pages visited by one user from the data set X to be clustered. The sessions and the contents of the pages visited by the user are used to generate the similarity matrix to be used for clustering the data set X. In other words, we use the pages visited by one user as a data set to test the clustering algorithm. This paper focuses on the clustering of one data set rather than the ensemble of multiple partitions. Each of the original logs contains several thousands of users (i.e., distinct IP addresses) and even more than hundred thousands of requests. After removing the requests which involve images, applets, .class, .exe files, and other irrelevant files, the number of users and the number of requests as well were reduced by a great deal. We observed that many users only visited just a few pages. In our experiments, we removed from the log files the users who visited less than three pages because the clustering to these pages does not contribute much to the evaluation of the performance of the clustering algorithm.

Table 1 summarizes the log data, where filel, file2, file3, and file4 refer to the log files 991001.gz, 99102.gz, 991003.gz, and 991004.gz, respectively. The data sets that are used to test the clustering algorithm are obtained from the logs that have been cleaned and do not contain the users who visited less than three pages.

In this experiment, the similarity of Web pages are measured based on the user access frequency as well as page contents. The access frequency can be obtained from analyzing the log files, while the best source of the page contents is the Web pages themselves. However, the actual Web pages are not available since there was no access to the source files of the Web site. For solving this problem, the topic vector for each page is constructed by making use of the URL information of that page appearing the log files. The URL portion of each visiting record in a log file indicates the path of the requested page in the Web site. In general, the name of the directory indicates the contents of the documents residing at that directory. Therefore, to some extent, the directory or subdirectory names appearing in the path of a page represent the contents of that page. Based on this intuition, we used the directory names for the path of a page as the topics of that page. For iinstance,"/lab/support/online/winNT4" is the path for a page requested in the logs, then "lab, support, online, winNT4" are the topics of the page.

TABLE 1. SUMMARY

Log Files	File1	File2	File3	File4
No.f req before preprocessing	121714	73467	76581	132924
No.f distinct users before preprocessing	5972	3699	4142	6861
No.f requests after preprocessing	5806	4281	5999	8098
No.f distinct users after preprocessing	1015	761	840	1455
No.f req after removing who req < 3 pages	4997	3665	5315	6888
No.f distinct users after removing users who req<3 pages	360	266	293	489

A. Performance Evaluation

In order to evaluate the performance of the algorithm, quantitative measures should be designed to measure the quality of the partition produced by the algorithm. According to the general definition of a clustering problem, a good clustering should maximize the similarity among the objects of each cluster (intra-similarity) and to minimize the similarity among objects of different clusters (inter-similarity). In other words, the distance between the objects within a cluster should be as small as possible (intra-distance), and the distance between clusters should be as large as possible (inter-distance). Hence, it is fairly acceptable and also convincing that the quality of a clustering can be evaluated from the two aspects, i.e., intradistance and inter-distance.

B. Results

Since the agglomerative single-link clustering algorithm (ASLCA) [3] has been widely used in various applications and considered as more versatile than many other algorithms, we decided to choose the ASLCA algorithm to

compare with the LCA algorithm proposed in this paper for the purpose of performance evaluation. The initial clusters included only one cluster which contains all the pages visited by the user. For the ASLCA, the cut-off of the similarity level (Threshold) has been set to 0.3.

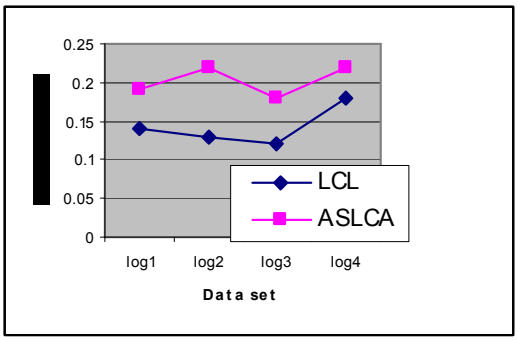

Fig. 4. Intra distance

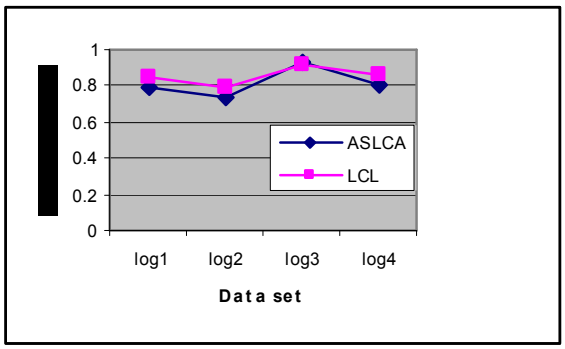

Fig. 5. Inter distance

Fig. 4 and Fig. 5 represent the average intra-distance and inter-distance of the clustering obtained by using both algorithms. From the results, we can see that the intra-distance produced by the LCL is lower than that produced by the ASLCA for all the four log files, and the inter-distance produced by the LCL is higher than that of the ASLCA except for the file3 file for which the LCL's result is only 0.01 higher than that of the ASLCA. Overall, the LCL outperforms the ASLCA.

C. Experiments

Fig. 6 shows the accuracy results of LCL and Partitional clustering algorithms. The LCL algorithm is compared with the Partitional clustering algorithm. In this experiment, we have considering 100 to 2500 different types of documents. Here LCL and Partitional Algorithms are applied to categorize the documents. It is observed from the figure that an LCL and Partitional Clustering algorithm gives same accuracy results up to 300 documents. When number of documents is increased, accuracy of the Partitional clustering algorithm is decreased. The accuracy of LCL algorithm is increased. Also the accuracy of LCL is more than partitional clustering algorithm. So, the accuracy of the individual clusters will be less than the accuracy of the Hybrid clustering is called LCL.

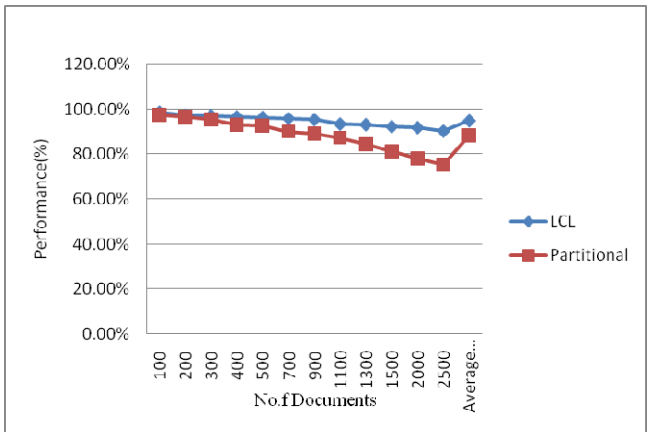

Fig. 6. Accuracy results of LCL and Partitional Clustering.

IV. CONCLUSION

In this paper, we propose a new hybrid algorithm which combines the features of leader's algorithm and complete-link algorithm. Using leader algorithm in the first level increases the speedness of the clustering and cohesion measure in the second level improves the performance of clustering. The time and the space complexities of LCL are also analyzed. Time complexity of this algorithm is linear to the size of the input data set. This algorithm leads to good clustering results while incurring a much shorter execution time.

REFERENCES

[l]. Barges and M. Levene. Data mining of user navigation patterns. In Proceedings of the Web Usage Analysis and User Profiling, volume I, pages 31-36, 1999.

[2]. P.Vijaya, M. N. Murthy and D.K. Subramanian. Leaders-Sub leaders: An efficient hierarchical clustering algorithm for large data sets. Pattern Recognition Letters 25

[3]. A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. ACM Computing Survey (CSUR), 31(3):264-323, 1999.

[4]. B. Mobasher. R. Cooley, and J. Srivastava. Automatic personalization based on web usage mining. Communication of ACM, 43(8). 2000.

[5]. M. Perkowitz and 0. Etzioni, Adaptive web sites: An ai challenge. 'In Proceedings of IJCAI-97. Volume 1, 1997.
[6]. M. Perkowitz and 0. Etzioni. Towards adaptive web sites: Conceptual framework and case study. Artificial Intelligence, 118245 - 275, 2000.
[7]. T. Toolan and N. Kusmerick. Mining web logs for personalized site maps. In Proceedings of the Intentional Conference on Web Information System Engineering, volume I, 2002.
[8]. ChengRuLin and Ming-Syan Chen A Robust and Efficient Clustering Algorithm based on Cohesion Self-Merging. In Proceedings of the Intentional Conference on SIGKDD '02 Edmonton, Alberta, Canada 2002 ACM.
[9]. Yue Xu Hybrid Clustering with Application to Web Mining. In Proceedings of the IEEE Intentional Conference 2005.