Evolutionary Clustering using Frequent Itemsets

Ravi Shankar Kiran G V R
International Institute of Information Technology
Hyderabad, India
{ravishankar.k,kiran.gvr}@research.iiit.ac.in

Vikram Pudi
International Institute of Information Technology
Hyderabad, India
vikram@iiit.ac.in

ABSTRACT

Evolutionary clustering is an emerging research area addressing the problem of clustering dynamic data. An evolutionary clustering should take care of two conflicting criteria: preserving the current cluster quality and not deviating too much from the recent history. In this paper we propose an algorithm for evolutionary clustering using frequent itemsets. A frequent itemset based approach for evolutionary clustering is natural and it automatically satisfy the two criteria of evolutionary clustering. We provide theoretical as well as experimental proofs to support our claims. We performed experiments on our approach using different datasets and the results show that our approach is comparable to most of the existing algorithms for evolutionary clustering.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

Keywords

Evolutionary Clustering, Frequent Itemsets, Data Streams

1. INTRODUCTION

Evolutionary clustering is an emerging research area addressing the problem of clustering dynamic data. Most of the existing clustering algorithms fail while handling such data. Lets consider the scenario of clustering a news feed. Hundreds of news articles get added to the data collection every day. We need a clustering algorithm that can generate separate clusterings at each timestamp, which at the same time also captures the essence of the entire data. A traditional clustering algorithm might not be able to perform such a task. An evolutionary clustering can solve such a problem by simultaneously optimizing two conflicting problems - (i) the current clustering should be of good quality (ii) the clustering should not deviate much from the previous clusterings.

StreamKDD'10, July 25, 2010, Washington, DC, USA

Evolutionary clustering has many practical applications to real world problems such as: finding out trends in the stock market, blog analysis to find out community development, social network evolution analysis, etc. When seen from a users perspective, evolutionary clustering produces clusters that smoothly evolve over time and hence the user will find familiarity in everyday's clustering. Due to such important applications, evolutionary clustering is being considered currently as an important area of research.

At a first look, the problem of evolutionary clustering might seem to be similar to clustering data streams [6, 1] and incremental clustering [6, 7]. Evolutionary clustering is similar to clustering data streams in that we deal with data that changes over time in both places. Also, in data stream clustering, we need to focus on optimizing time and space constraints. But in evolutionary clustering, we focus on how to obtain a clustering that evolves smoothly over time. On the other hand, incremental clustering also concentrates on computational efficiency at the cost of low cluster quality. Also, in incremental clustering, the new clustering might not be related to the existing clustering. This is not the case with evolutionary clustering because it is entirely based on the concept of maintaining the clustering over time.

A recent trend in clustering huge text data is the use of frequent itemsets. These methods handle the high dimensionality of the data by considering only the items which are frequent for clustering. A frequent itemset is a set of words which occur together frequently and are good candidates for clusters. By considering only the items which occur frequently in the data, we can also address problems like outlier removal, dimensionality reduction, etc. In addition to these, frequent itemsets naturally satisfies the main features of evolutionary clustering mentioned in [3] - Consistency, Noise Removal, Smoothness of Evolution and Cluster Correspondence. We explain this in detail in Section 4.2.

2. PROBLEM DEFINITION

In this paper, we propose an algorithm for solving the problem of evolutionary clustering using frequent itemsets. We argue that using frequent itemsets is the best method to solve the problem of evolutionary clustering. We prove our argument experimentally by the results provided in Section 6.

Let $D = (D_1, D_2, D_3, ..., D_n)$ be sets of documents at different timestamps, where $D_i = (d_1, d_2, ..., d_k)$ is the set of documents at timestamp t_i. Let C_i be the clustering of all the data until timestamp t_i. Now, when the data D_{i+1} arrives at timstamp t_{i+1}, our evolutionary clustering algorithm

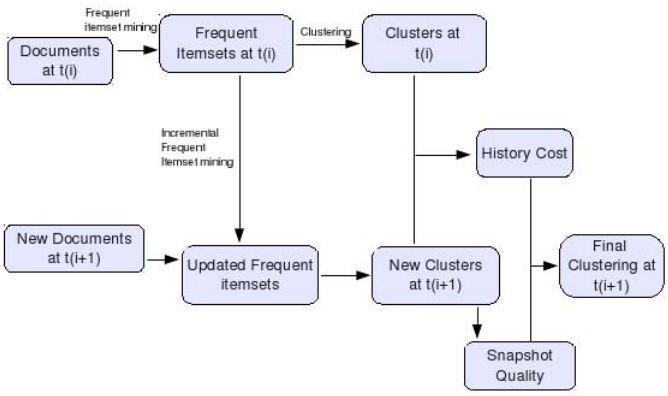

Figure 1: Overview of our architecture

produces a clustering C_{i+1} such that two distinct criteria are met: the history cost must be low, meaning that the clustering should not deviate much from the previous clustering and the snapshot quality should be high, meaning that the clustering quality should be good.

Figure 1 shows our architecture in detail. We use frequent itemset based methods for clustering. Using frequent itemsets for clustering stream data makes the clustering simpler because, when we have data at a particular timestamp and new data arrives, we just have to perform an incremental frequent itemset mining to obtain the new clusters. We need not have the entire data of the previous timestamps because the frequent itemsets at those timestamps can be used to represent the entire data.

Our focus here is on an online solution, that is, we do not have information about data at time t+1 while we are at timestamp t. Clearly we can see that a clustering produced in an online setting may not be as good as the offline solution, but solutions to real world problems need an online approach. In the rest of the paper, we use the terms frequent itemset, cluster and topic interchangeably.

The rest of the paper is organized as follows: Section 3 talks about the related work. Section 4 briefly describes the background needed, Section 5 describes our approach in detail. Section 6 presents the experiments that support our approach. Finally we conclude our paper and present the future work in Section 7.

3. RELATED WORK

In this section, we detail the work that has been done in the fields of evolutionary clustering and clustering using frequent itemsets.

Evolutionary clustering is a new research topic first formulated by Chakrabarti, et al in [3]. In their work, they developed a framework where they define the two properties of evolutionary clustering: 1. snapshot quality, which indicates the quality of the clustering at the current timestamp; 2. history cost which ensures that the clusters evolve smoothly over time. The framework attempts to combine these two properties and achieve a good clustering. They developed two algorithms based on this framework: evolutionary k-means and evolutionary hierarchical clustering. For each of these algorithms, they define functions for snapshot quality and history cost.

Another recent work in evolutionary clustering was by Chi, et al [4] in which they propose an evolutionary spectral clustering approach by including the parameter temporal smoothness to solve the problem. They used the temporal smoothness parameter to achieve a clustering that evolves smoothly with time. Based on this, they proposed two algorithms PCQ(Preserving Cluster Quality) and PCM(Preserving Cluster Membership). These two algorithms try to incorporate clustering history information into traditional algorithms like spectral clustering, k-means, hierarchical clustering, etc. In PCQ, the current partition is applied to historic data and the resulting cluster quality is defined as the temporal cost(history cost). In PCM, the current partition is compared to the historic partition and the resulting difference determines the history cost. But this paper has a major draw back that they assume that the number of clusters remain constant over time. For many cases in an evolutionary clustering scenario, this assumption is violated.

Xu, et al [11] propose a statistical solution to the problem of evolutionary clustering. They use Dirchlet process to model the evolutionary change of the clusters over time. They propose two approaches to solve the problem: DPChain and HDP-EVO. DPChain is based on Dirchlet Process Model, in which the cluster mixture proportion information at different times is used to reflect the smooth cluster change over the time. HDP-EVO is based on the Hierarchical Dirchlet Model, which uses a hierarchy of clusters to address the cluster correspondence issue in order to solve the evolutionary clustering problem.

The most recent work in this area is by Xu, et al [12] which combines Hierarchical Dirchlet Process used above in [11] and Hierarchical Transition Matrix based on Infinite Hierarchical Hidden Markov State model to bring out an effective solution to the problem of evolutionary clustering.

In recent times research is being done on clustering using frequent itemsets. This type of clustering is different from traditional clustering algorithms in that it uses only the frequent words for clustering. Frequent Term based Clustering (HFTC) [2] has been the first algorithm in this regard. But HFTC was not scalable and Fung, et al came up with Hierarchical Document Clustering using Frequent itemsets(FIHC) [5] which outperforms HFTC. It provides a hierarchical clustering with labels to the clusters. Some of the drawbacks of FIHC include (i)using all the frequent itemsets to get the clustering (number of frequent itemsets may be very large and redundant) (ii) Not comparable with previous methods like UPGMA(Unweighted Pair Group Method with Arithmetic Mean) [8] and Bisecting K-means [14] in terms of clustering quality. (iii) Use hard clustering (each document can belong to at most one cluster), etc. Then Yu, et al came up with a much more efficient algorithm using closed frequent itemsets for clustering (TDC) [13]. They also provide a method for estimating the support correctly. But they use closed itemsets which also may be redundant. Recently Hasan H Malik, et al. proposed Hierarchical Clustering using Closed Interesting Itemsets(HCCI) [9], which is the current state of the art algorithm in clustering using frequent itemsets. In this, they introduce the notion of closed interesting itemsets to be those itemsets which are interesting based on certain interestingness measures like Chi-Square, Cosine, Gini Index, etc.

4. BACKGROUND

In this section, we try to establish the background required for our work.

4.1 Frequent itemset based document clustering

Intuitively, the document clustering problem is to cluster text documents by using the idea that *similar documents share many common keywords*. Keywords in documents are treated as *items* and the documents (being treated as sets of keywords) are analogous to transactions in a market-basket dataset. This forms a transaction space, that we refer to as **doc-space** as illustrated below, where the d_i's are documents and w_{ij}'s are keywords.

d_1 – $[w_{11}, w_{21}, w_{31}, w_{41}, \ldots]$
d_2 – $[w_{12}, w_{22}, w_{32}, w_{42}, \ldots]$
d_3 – $[w_{13}, w_{23}, w_{33}, w_{43}, \ldots]$
...

Then, in this doc-space, frequent combinations of keywords (i.e., frequent itemsets) that are common to a group of documents convey that those documents are similar to each other and thereby help in defining clusters. e.g: if (a, b, c) is a frequent itemset of keywords, then (d_1, d_3, d_4) which are the documents that contain these keywords form a cluster. We refer to the set of documents containing a frequent itemset as the *doc-list* of that frequent itemset.

Consider data at a timestamp t_i. Applying frequent itemset mining algorithm on that data, let FS(i) be the set of frequent itemsets. Let doc(i) be the set of documents corresponding to the i^{th} frequent itemset in FS(i). Using the justification provided above, we consider every such set of documents (doc(i)) thus obtained as an initial cluster. But with this method, there are a lot of overlaps between clusters, i.e. a single document can belong to multiple frequent itemsets and thus can belong to multiple clusters. To reduce the overlaps between clusters, we use the score function similar to that proposed by Yu, et al in [13]

$$Score(d, T) = \sum_{t \in T} (d \times t) \,/\, length(T) \qquad (1)$$

where $d \times t$ denotes the tf-idf score of each word t in T, the frequent itemset in document d.

Using Eqn. 1, we calculate the score of a document in a cluster and assign a document to the cluster with the highest score. In order to allow a document to belong to multiple clusters, we put a user given threshold (MAX_DUP) and assign each document to atmost MAX_DUP number of clusters. Thus we have the clusters at a timestamp t(i).

4.2 Why we use frequent itemsets for evolutionary clustering

Generally any frequent itemset must satisfy one of the following criteria. Let F be the frequent itemset.

(i) Support(F) $\ll$ Min_sup
(ii) Support(F) $\simeq$ Min_sup
(iii) Support(F) $\gg$ Min_sup

, where Min_sup is the minimum support threshold for frequent itemset mining. In case (i), the frequent itemset does not have a support greater than minimum support and so is of no importance. In case (iii), the frequent itemset has a large support, so even if there are minor variations in the support of that frequent itemset over time, the clustering is not much affected. The frequent itemsets in case (ii) are of importance because they play an important role in changing the clustering significantly. But in our case, since each document belongs to only those frequent itemsets(clusters) in which it has the highest score, the number of documents contained in these frequent itemsets will be very small because the documents will already be present in those frequent itemsets in case (iii)(because they have a higher supports, and hence higher scores). Thus, most of such frequent itemsets become empty and hence do not effect the clustering.

Now, we try to explain how frequent itemsets automatically handles the four main features of evolutionary clustering defined in [3] - Consistency, Noise Removal, Smoothess of evolution and Cluster correspondence.

Consistency - The clustering at any timestamp should be consistent with the clustering at a previous timestamp, i.e. the clustering should not change drastically with time. In our case, when data at a new timestamp gets added to the existing data, there will not be much change in the existing frequent itemsets and the number of frequent itemsets that are newly added is very small. e.g. If (a,b,c) is an existing frequent itemset, the chance of (a,b,c) becoming (a,b,c,d) is more than an entirely new itemset (x,y,z) becoming frequent. Even if such new frequent itemsets appear, they fall into case(ii) explained above. So the effect of such new frequent itemsets on the clusters will not be significant. Since the frequent itemsets do not change much, the clustering will be consistent.

Noise Removal - Our method automatically takes care of noise because the noise containing documents will not be frequent. So, there is no chance of them being added to our clusters.

Smoothness of evolution - If the clusters shift over time, evolutionary clustering must represent this change. Using frequent itemsets, if there is a smooth shift in the clusters over time, the support of such documents increases gradually and reaches the minimum support level, when it becomes frequent and gets added to the set of clusters. Hence, smoothness is guaranteed by using frequent itemsets.

Cluster Correspondence - In an evolutionary clustering, one of the most difficult and challenging issues is to solve the correspondence problem. The correspondence problem refers to finding a correspondence between clusters of the current timestamp and the previous timestamp due to the evolution of the clustering. In our case, as described in the case of consistency, the frequent itemsets do not change much and so it is easy to find a frequent itemset in the current timestamp corresponding to one in the previous timestamp.

5. OUR APPROACH

In this section, we describe our approach in detail. Figure 1 explains our approach. At an initial timestamp t_i, we use the approach described in Section 4.1 to cluster the documents to get the initial clusters C_i. Now, when data at the next timestamp(t_{i+1}) arrives, we use an incremental frequent itemset algorithm to update the frequent itemsets and thus obtain the clusters(C_{i+1}).

5.1 Incremental frequent itemset mining

At timestamp t_{i+1}, when a new document set arrives, we

apply an incremental frequent itemset mining algorithm on the frequent itemsets at the previous timestamp and the current data to obtain the new frequent itemsets of the entire data. As explained above in Section 4.1, each of the newly obtained frequent itemset is a potential cluster.

Here four cases may arise:

1. An itemset which was frequent earlier becomes infrequent now and there exists a cluster corresponding to that itemset in the previous clustering (timestamp).

2. A new frequent itemset being created.

3. A frequent itemset in the previous timestamp remaining frequent currently too, but a document in an older cluster might have a higher score for a new cluster than the older one.

4. An itemset which was infrequent previously remains infrequent.

We leave out the last case as we are not interested in infrequent itemsets and concentrate on the remaining three cases.

- When a frequent itemset becomes infrequent, we compute the scores of each document in the doc-list of this frequent itemset with the newly obtained frequent itemsets and reassign each document to MAX_DUP clusters for which the document has the highest score. We then reduce the overlapping between clusters using Eqn. 1. If none of the documents remain in the cluster, we prune the cluster.

- When a new frequent itemset gets created, for each document in its doc-list we compute the score with all the other frequent itemsets it belongs to. This also contains the case where a document in an older cluster might have a higher score for a newly obtained cluster(case 3 above). This problem is automatically solved by our approach, because when we compute the score of an older document with a new one, we assign it to the newer one if its score is higher.

After resolving the above issues, we obtain the updated clusters at timestamp t_{i+1}. Now we compute the Snapshot quality of these updated clusters:

Snapshot quality: We define the snapshot quality at any timestamp to be the quality of the clustering indicated by general measures like F-Score [14], NMI(Normalized Mutual Information) [10], etc.

Using the clusterings C_i and C_{i+1}, we compute the history cost as follows:

History Cost: History cost indicates the deviation of the current clustering from the previous clustering. To measure the history cost, we define the following function:

Let $f : [k] \rightarrow [k]$ be the function which maps the clustering at t_i to the clustering at t_{i+1}, i.e. we match each frequent itemset in C_i with the best frequent itemset in C_{i+1} using the function f. For each $c^k \in C_{i+1}$, we find the corresponding $c^{f(k)} \in C_i$ using the function f. Then, we compute the doc-list of each of these frequent itemsets. Let A and B be the two doc-lists for two frequent itemsets $c_i \in C_i$ and $c_{i+1} \in C_{i+1}$ respectively. We compute the set S' = (A-B) $\cup$ (B-A). Here, S' represents the change in the documents between the corresponding clusters.

Now, the history cost is defined as,

$$hc(c_i, c_{i+1}) = \sum_{s \in S'} (Score(s, T)) \quad (2)$$

where $Score(s, T)$ indicates the scores of documents in their respective frequent itemsets calculated using Eqn. 1 and T is defined as follows: $if\ s \in c_i$, $T = c_{i+1}$, $else\ if\ s \in c_i$, $T = c_i$.

Now, the total history cost at timestamp t_{i+1} is defined as

$$HC(C_i, C_{i+1}) = \sum_{c_i \in C_i, c_{i+1} \in C_{i+1}} hc(c_i, c_{i+1}) \quad (3)$$

6. EXPERIMENTAL EVALUATION

In this section we detail the various experiments we have done to prove the validity of our approach. We compare our approach with evolutionary k-means and other recent approaches for evolutionary clustering using various metrics like F-Score [14] and NMI [10].

To evaluate our algorithm on cluster quality and compare it with different algorithms, we used 2 famous datasets - (i) Reuters 21578 dataset [1], (ii) 20 News Group Dataset [2]. We applied general preprocessing techniques like stop word removal, stemming, etc on all the documents in these datasets. For calculating NMI and F-score using the 20 News Group dataset, we used the same method used in [11] for selecting the documents - we selected 10 topics from the dataset (comp.graphics, rec.autos, rec.sport.baseball, sci.crypt, sci.electronics, sci.med, sci.space, soc.religion.christian, alt.atheism, talk.politics.mideast) with each having 100 documents. To simulate 5 different timestamps, we divided the data into 5 parts, each having 20 documents randomly selected from these clusters. For calculating F-Score on Reuters dataset, we divided the dataset into 5 parts randomly to indicate 5 different timestamps. We clustered the documents at each timestamp using evolutionay k-means and our algorithm and computed the F-Score. The results for this are shown in Table 2. For computing the history cost in Figure 3, we divided the 20 News Group dataset into 10 timestamps in the same way described above.

Tables 1 and 2 show that our method outperforms evolutionary k-means on both the datasets. Also, as time progresses, the variation in F-score(cluster quality) of our method is very low when compared to evolutionary k-means. This indicates that our algorithm achieves smoothness of evolution. Also, we can see from Table 2 that the values of F-score for k-means is fluctuating badly. We suspect this might be due to noise or unimportant topics being introduced in the new timestamps. But our method handles these variations smoothly and produces smooth clusters which can be seen from minute variations in the F-score.

Figure 2 compares our method with other recent methods for evolutionary clustering based on NMI performance. We can see that our method performs better than PCQ and PCM [4] and comparable to DPChain and HDPEVO [11]. This is a very satisfying result because, as shown in Section 4.2 our method is best suited for solving the problem of evolutionary clustering and also achieves consistent results.

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

[2] http://kdd.ics.uci.edu/databases/20newsgroups

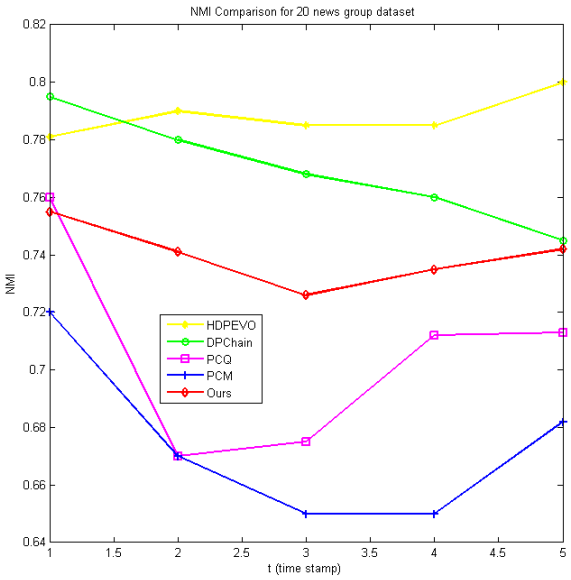

Figure 2: Comparison of NMI performance of different algorithms on 20 News Group dataset

Table 1: Comparison of F-Score on 20 News Group dataset

Timestamp	Evolutionary K Means	Ours
1	0.57	0.72
2	0.42	0.62
3	0.33	0.60
4	0.29	0.56
5	0.17	0.53

Figure 3 shows a graph of history cost plotted against time. We can see that as time progresses the history cost almost remains constant. This is also an indication of smoothness of evolution of our clusters.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we provided a method to perform evolutionary clustering using frequent itemsets. We also showed that using frequent itemsets for evolutionary clustering is quite natural and would lead to good results. Our results in Section 6 show that our method is comparable to the recent methods in evolutionary clustering [11] and much better than the traditional approaches like evolutionary k-means [3].

Using closed and generalized frequent itemsets for clustering might help us reduce the dimensionality even more. We plan to look into how these methods help us with improving the quality of the evolutionary clustering. We also plan to use external knowledge sources like Wikipedia, WordNet, etc to enhance the document representations so as to achieve better clustering quality and lower history cost.

Table 2: Comparison of F-Score on Reuters-21578 dataset

Timestamp	Evolutionary K Means	Ours
1	0.448	0.801
2	0.372	0.797
3	0.556	0.784
4	0.412	0.778
5	0.525	0.787

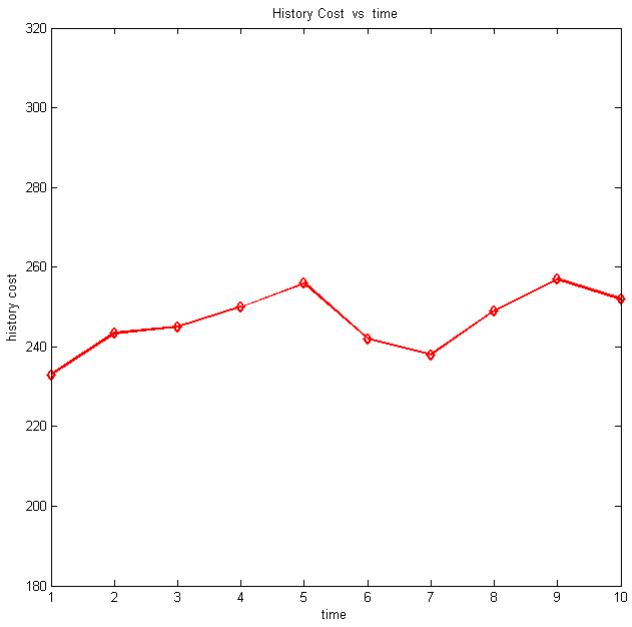

Figure 3: History Cost vs. Time for 20 News Group dataset

8. REFERENCES

[1] C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *Proceedings of 12th Very Large Databases Conference*, 2003.

[2] F. Beil, M. Ester, and X. Xu. Frequent Term-based Text Clustering. In *Proceedings of International Conference on Knowledge Discovery and Data Mining*, 2002.

[3] D. Chakrabarti, R. Kumar, and A. Tomkins. Evolutionary Clustering. In *Proceedings of Knowledge Discovery and Data Mining(KDD'06)*, August 2006.

[4] Y. Chi, X. Song, D. Zhou, K. Hino, and B. L. Tseng. Evolutionary Spectral Clustering by Incorporating Temporal Smoothness. In *Proceedings of Knowledge Discovery and Data Mining(KDD '07)*, August 2007.

[5] B. Fung, K. Wang, and M. Ester. Hierarchical Document Clustering using Frequent Itemsets. In *Proceedings of SIAM International Conference on Data Mining*, 2003.

[6] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams. In *IEEE Symposium on Foundations of Computer Science*, 2000.

[7] C. Gupta and R. Grossman. Genic: A single pass generalized incremental algorithm for clustering. In *Proceedings of SIAM Int. Conf. on Data Mining*, 2004.

[8] L. Kaufman and P. J. Rousseeuw. Finding Groups in Data, An introduction to Cluster Analysis. New York John Wiley & Sons, Inc, March, 1990.

[9] H. H. Malik and J. R. Kender. High Quality, Efficient Hierarchical Document Clustering Using Closed Interesting Itemsets. In *Proceedings of International Conference on Data Mining (ICDM'06)*, 2006.

[10] A. Strehl and J. Ghosh. Cluster Ensembles a knowledge reuse framework for combining partitions. In *Proceedings of AAAI*, 2002.

[11] T. Xu, Z. M. Zhang, P. S. Yu, and B. Long. Dirichlet Process Based Evolutionary Clustering. In *Proceedings of International Conference on Data Mining(ICDM) 2008*, 2008.

[12] T. Xu, Z. M. Zhang, P. S. Yu, and B. Long. Evolutionary Clustering by Hierarchical Dirichlet Process with Hidden Markov State. In *Proceedings of International Conference on Data Mining(ICDM) 2008*, 2008.

[13] H. Yu, D. Searsmith, X. Li, and J. Han. Scalable Construction of Topic Directory with Nonparametric Closed Termset Mining. In *Proceedings of International Conference on Data Mining (ICDM'04)*, 2004.

[14] Y. Zhao and G. Karypis. Evaluation of Hierarchial Clustering Algorithms for Document Datasets. In *Proceedings of International Conference on Information and Knowledge Management*, November 2002.