Review on Frequent Item-Based Dynamic Text Clustering

Dr. M. Usha Rani
Associate Professor, Department of Computer Science, Sri Padmavati Mahila Visva Vidyalayam,
Tirupati -517502, AP, INDIA.

Abstract— **In today's information age, the significance of text clustering is emergent all the time with the incredible non-stop generation of massive volumes of unstructured data. Fast and high-quality text clustering algorithms play a vital role towards the goal of document organization among developed methods. In general, data always arrives in a multiple, continuous, rapid and time varying flow. Evolutionary clustering is an emerging research area to address the problem of clustering such 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. Incremental hierarchical text document clustering algorithms are important in organizing documents generated from streaming on-line sources. Text clustering in accordance with the frequent itemsets provides significant dimensionality reduction. In addition, frequent itemsets address the problems like outlier removal, dimensionality reduction, etc. and automatically satisfy the main criteria of evolutionary clustering, naturally. Thus, a frequent item based approach is appropriate /suitable for evolutionary clustering. In evolutionary clustering, the focus is 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. In incremental clustering, the new clustering might not be related to the existing clustering whereas, evolutionary clustering is entirely based on the concept of maintaining the clustering over time.**

Several researches revealed that the frequent item-based dynamic text clustering or evolutionary clustering balances both the current and historical behavior of data. Using closed and generalized frequent itemsets for clustering may reduce the dimensionality further, thus improving the quality of evolutionary clustering. Exploiting an effective and efficient method in evolutionary text clustering to achieve better clustering quality at lower history cost would be an essential direction for research.

Keywords - Text Clustering; Frequent Itemset; Incremental Clustering; Evolutionary Clustering

I. Introduction

Owing to the current explosion of information and the accessibility of cheap storage, collecting enormous data has been achievable during the last decades. The ultimate intent of this massive data collection is the utilization of this information to achieve competitive benefits, by determining formerly unidentified patterns in data that can direct the process of decision making. The analysis of data with the aid of Online Analytical Processing (OLAP) tools alone is highly tedious; thus necessitating an automated process to ascertain interesting and concealed patterns in data. Data mining is formally defined as "A process of non-trivial extraction of implicit, previously unknown and potentially useful information from the data stored in a database" [6]. Generally, data mining is performed on data represented in quantitative, textual, or multimedia forms. Text Mining is the discovery of new, previously unknown information by automatically extracting information from text documents. It is an increasingly important research field because of the necessity of obtaining knowledge from the enormous number of text documents [16]. Text clustering is one of the fundamental functions in text mining. Text clustering is to group a collection of documents (unstructured texts) into different category groups so that documents in the same category group describe the same subject. Fast and accurate clustering of documents plays an important role in the field of text mining and automatic information retrieval systems.

Text clustering is an effective tool to manage information overload. By clustering similar documents together, large document collections can be quickly browsed, distinct topics and subtopics (concept hierarchies) in them can be easily grasped, and they can be efficiently queried among many other applications. Hence, it has been widely studied. Clustering of text documents plays a vital role in efficient document organization, summarization, topic extraction and information retrieval. It was initially used for improving the precision or recall in an information retrieval system. Recently, clustering has been proposed for use in browsing a collection of documents or in organizing the results returned by a search engine in response to a user's query. Document clustering has also been used to automatically generate hierarchical clusters of documents.

In general, data always arrives at a database in a multiple, continuous, rapid and time varying flow. So, document databases now face the major issue of high update rates. In such environment, applications of clustering need to frequently perform complete and costly re-clustering. Researchers have found that existing clustering algorithms are not appropriate for preserving the clusters in such dynamic environment, and they have recognized the problem of updating the clusters without habitually performing the complete re-clustering of the entire database, each time when a change is made in the database. To overcome this disadvantage, a proficient approach emerged for clustering the dynamic databases.

Most of the document clustering algorithms studied in the literature operates in a batch mode, i.e., they collect all the documents to be clustered before the start of the exercise and cluster them by making multiple iterations over them. But, this is not always feasible. With the advent of online publishing in the World Wide Web, the number of documents being generated everyday has increased considerably. Popular sources of informational text documents such as Newswire and Blogs generate documents virtually continuously. To organize such documents using existing batch clustering algorithms one must cluster documents collected up to certain points in time. This would lead to repeated processing of older documents, the large majority of which do not change cluster membership. This would result in redundancy that would make the exercise extremely time consuming, due to the volume of documents being generated. This problem can be addressed by batch clustering periodically collected small batches of documents and then merging the generated clusters. However, there will always be a wait time before a newly generated document can appear in the cluster hierarchy. This delay would be unacceptable in several important scenarios, e.g., financial services, where trading decisions depend on breaking news, and quick access to appropriately classified news documents is important. A clustering algorithm in such a setting needs to process the documents as soon as they arrive. This calls for the use of an *incremental* or *online* or *evolutionary* clustering algorithm. This is a relatively less explored area in text document clustering literature.

A. Evolutionary Clustering vs. Incremental Clustering

Clustering of text documents can be done either in an evolutionary fashion or incrementally. Evolutionary clustering might seem to be similar to clustering data streams [10, 1] and incremental clustering [10, 11]. Evolutionary clustering is similar to clustering data streams that deal with data that change over time. Also, in data stream clustering, the focus is on optimizing time and space constraints. But in evolutionary clustering, the focus is 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. An incremental algorithm can cluster text documents into an informative cluster hierarchy as they arrive. In incremental clustering, a model is incrementally updated as new data arrive, primarily to avoid the cost of storing all historical similarities [8]. However, in evolutionary clustering, the focus is upon optimizing a new quality measure which incorporates deviation from history. An evolutionary clustering algorithm is online if it must provide a clustering of the data during timestep t before seeing any data for timestep t+1.

B. Need and Importance of Evolutionary Clustering

Evolutionary clustering is an emerging research area that addresses the problem of clustering dynamic data. Most of the existing clustering algorithms fail while handling such data. Consider the scenario of clustering a news feed. Hundreds of news articles get added to the data collection every day. A clustering algorithm is needed 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.

Here is an illustration to show why the history has a significant impact on the clustering sequence. Consider a data set in which either of two features may be used to split the data into two clusters: feature A and feature B. Each feature induces an orthogonal split of the data, and each split is equally good. However, on odd-numbered days, feature A provides a slightly better split, while on even-numbered days, feature B is better. The optimal clustering on each day will shift radically from the previous day, while a consistent clustering using either feature will perform arbitrarily close to optimal. In this case, a clustering algorithm that does not take history into account will produce a poor clustering sequence.

Evolutionary clustering is the problem of processing timestamped data to produce a sequence of clusterings; that is, a clustering for each timestep of the system. Each clustering in the sequence should be similar to the clustering at the previous timestep, and should accurately reflect the data arriving during that timestep. The primary setting for this problem is the following. Every day, new data arrives for the day, and must be incorporated into a clustering. If the data does not deviate from historical expectations, the clustering should be "close" to that from the previous day, providing the user with a familiar view of the new data. However, if the structure of the data changes significantly, the clustering must be modified to reflect the new structure. Thus, the clustering algorithm must trade off the benefit of maintaining a consistent clustering over time with the cost of deviating from an accurate representation of the current data. The benefits of evolutionary clustering compared to traditional clustering appear in situations in which the current clustering is being consumed regularly by a user or system. In such a setting, evolutionary clustering is useful for the following reasons:

(1) Consistency: A user will find each day's clustering familiar, and so will not be required to learn a completely new way of segmenting data. Similarly, any insights derived from a study of previous clusters are more likely to apply to future clusters.

(2) Noise removal: Providing a high-quality and historically consistent clustering provides greater robustness against noise by taking previous data points into effect. As we describe later, our method subsumes standard approaches to windowing and moving averages.

(3) Smoothing: If the true clusters shift over time, evolutionary clustering will naturally present the user with a smooth view of the transition.

(4) Cluster correspondence: As a side effect of the framework, it is possible to place today's clusters in correspondence with yesterday's clusters. Thus, even if the clustering has shifted, the user will still be situated within the historical context.

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.

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, address problems like outlier removal, dimensionality reduction, etc. can also be addressed. In addition to these, frequent itemsets naturally satisfies the main features of evolutionary clustering mentioned in [3, 5] – Consistency, Noise Removal, Smoothness of Evolution and Cluster Correspondence.

II. FREQUENT ITEM-BASED DYNAMIC TEXT CLUSTERING

Evolutionary clustering is an emerging research area of text clustering that deals with the clustering of dynamic data. It is the problem of processing timestamped data to produce a sequence of clusterings. An evolutionary clustering algorithm must produce a sequence of clusterings, one for each timestep. The clustering produced during a particular timestep should measure well along two distinct criteria: it should have low history cost, meaning it should be similar to the previous clustering in the sequence, and it should have high snapshot quality, meaning it should be a high-quality clustering of the data that arrived during the current timestep. The snapshot quality reflects the underlying figure of merit driving a particular clustering algorithm in a non-evolutionary setting and the history cost must allow a comparison of clusterings, in order to determine how much the later one has shifted from the earlier. This comparison must address the issue that some data will appear for the first time in the later clustering, while some data will be seen in the earlier clustering but not the later one, and so forth. Thus, in order to perform well, clustering must include the objects that have been seen in the past but have not appeared during the current timestep. So, an evolutionary clustering algorithm must "carry along" information about the appropriate location of historical information, either implicitly or explicitly.

A. *Why to use Frequent Itemsets for Evolutionary 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, address problems like outlier removal, dimensionality reduction, etc. can also be addressed.

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

(i) Support (F) << Min_sup

(ii) Support (F) $\underset{\sim}{}$Min_sup

(iii) Support (F) >> 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, 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.

Frequent itemsets automatically handles the four main features of evolutionary clustering - Consistency, Noise Removal, Smoothness of evolution and Cluster correspondence [3].

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 case of frequent itemset evolutionary clustering, 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 – This 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 the 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 this 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.

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 are treated as sets of keywords. This forms a transaction space, referred to as doc-space as illustrated below:

d1 - [w_{11}; w_{21}; w_{31}; w_{41} . . .]

d2 - [w_{12}; w_{22}; w_{32}; w_{42} . . .]

d3 - [w_{13}; w_{23}; w_{33}; w_{43} . ..]

. . .

where the d_i's are documents and w_{ij} 's are keywords. 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 (d1; d3; d4) which are the documents that contain these keywords form a cluster. The set of documents containing a frequent itemset is referred to 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, every such set of documents (doc (i)) thus obtained is considered 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, the score function proposed by Yu, et al in [13] may be used, and the score of a document in a cluster is calculated and then a document is assigned to the cluster with the highest score. In order to allow a document to belong to multiple clusters, a user given threshold (MAX DUP) is put and each document is assigned to at most MAX DUP number of clusters. Thus, one can have the clusters C_i at an initial timestamp t_i. When a new data or document set arrives at the next timestamp (t_{i+1}), an incremental frequent itemset mining algorithm can be applied on the frequent itemsets at the previous timestamp and the current data to obtain the new frequent itemsets of the entire data. Each of the newly obtained frequent itemset is a potential cluster and thus, the frequent itemsets are updated and the clusters (C_{i+1}) can be obtained.

III. REVIEW OF RELATED LITERATURE

In data mining literature, a handful of researches for clustering the text data exist [17] [22] [7] [23] [14] based on the popular clustering algorithms (partitional and hierarchical clustering) and amongst them now-a-days, the concentration of recent researchers is focused on frequent item set based text clustering. A review of researches and the work that has been done in the fields of evolutionary clustering and clustering using frequent itemsets is furnished in this section.

Evolutionary clustering is a new research topic first formulated by Chakrabarti, et al in [3]. In their work, they developed a framework where they defined the two properties of evolutionary clustering: 1. snapshot quality, which indicates the quality of the clustering at the current times tamp; 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 defined functions for snapshot quality and history cost.

Another recent work in evolutionary clustering was by Chi, et al [4] in which they proposed 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 attempted 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 drawback that they assumed that the number of clusters remain constant over time. For many cases in an evolutionary clustering scenario, this assumption is violated.

Xu, et al [18] proposed a statistical solution to the problem of evolutionary clustering. They used Dirchlet process to model the evolutionary change of the clusters over time. They proposed 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 [19] which combines Hierarchical Dirchlet Process used in [18] 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) [20] 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 item sets 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) [12] and Bisecting K-means [21] 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) [20]. They also provided a method for estimating the support correctly. But, they used closed itemsets which also may be redundant. Recently, Hasan H Malik, et al. proposed Hierarchical Clustering using Closed Interesting Itemsets (HCCI) [13], which is the current state of the art algorithm in clustering using frequent itemsets. In this, they introduced 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.

Murali Krishna et al [14] developed a proficient approach for clustering dynamic data using frequent itemsets. The proposed algorithm consists of two phases: Primary Clustering and Incremental Clustering. In the primary clustering, the documents in the static database are preprocessed, and then the incremental text clustering approach is carried out. The incremental database is clustered by making use of initial clusters and frequent keywords. The frequent keywords of the incoming text document from the incremental database were used for finding the relevant partition and in accordance with the similarity measure; the appropriate cluster was identified for the incoming text document. The experimental results showed that the proposed incremental clustering is efficient in terms of precision, recall and F-measure.

Ravi Shankar, et al [15] proposed an algorithm for solving the problem of evolutionary clustering using frequent itemsets. Using frequent itemsets for clustering stream data makes the clustering simpler because, when data at a particular timestamp is available and new data arrives, then it is enough to perform an incremental frequent itemset mining to obtain the new clusters. There is no need to have the entire data of the previous timestamps because the frequent itemsets at those timestamps can be used to represent the entire data. When a new document set arrives, an incremental frequent itemset mining algorithm is applied on the frequent itemsets at the previous timestamp and the current data to obtain the new frequent itemsets of the entire data.

IV. CONCLUSION

In conclusion, the importance of text clustering will continue to grow along with the massive volumes of unstructured data generated. Exploiting an effective and efficient method in text clustering would be an essential direction for research in text clustering. Evolutionary clustering is an emerging research area addressing the problem of clustering dynamic data. An evolutionary clustering framework requires that the clustering at any point in time should be of high quality while ensuring that the clustering does not change dramatically from one time step to the next. A frequent itemset based approach for evolutionary clustering is natural and it automatically satisfy the above two criteria of evolutionary clustering. The work that has been done in the fields of evolutionary clustering and clustering using frequent itemsets is still in its infancy.

V. FUTURE WORK

Dynamic text clustering is a relatively less explored area in text document clustering literature. The use of frequent association for dynamic text clustering has received a great deal of attention in research communities since the mined frequent itemsets reduces the dimensionality of the documents drastically. Using closed and generalized frequent itemsets for clustering might help to reduce the dimensionality even more, thus improving the quality of evolutionary clustering. External knowledge sources like Wikipedia, WordNet, etc can be used to enhance the document representations so as to achieve better clustering quality and lower history cost.

An evolutionary clustering algorithm is online if it must provide a clustering of the data during timestep t before seeing any data for timestep t+1. If the algorithm has access to all data beforehand, it is offline. Offline algorithms may “see the future,” and should perform at least as well as their online counterparts. However, the online setting is more important for real-world applications, and there are no natural, efficient offline algorithms that may be easily employed as lower bounds. Therefore, there is a great scope to consider the offline problem for future work.

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] Deepayan Chakrabarti. Clustering Applications at Yahoo!, In NIPS 2009 Workshop on Clustering.

[6] Fayyad U, “Data Mining and Knowledge Discovery in Databases: Implications from scientific databases,” In Proc. of the 9th Int. Conf. on Scientific and Statistical Database Management, Olympia, Washington, USA, pp. 2-11, 1997.

[7] Feldman R., Sanger J., “The Text Mining Handbook”, Cambridge University Press, 2007.

[8] D. Fisher. Knowledge acquisition via incremental conceptual clustering. Machine Learning, 2:139–172, 1987.

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

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

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

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

[13] 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.

[14] Murali Krishna S., Durga Bhavani S., “An Efficient Approach For Text Clustering Based On Frequent Itemsets”, ©EuroJournals Publishing, Inc. 2010, European Journal of Scientific Research ISSN 1450-216X, Vol.42, n 3, pp. 399-410, 2010.

[15] Ravi Shankar, Kiran G V R, and Vikram Pudi. Evolutionary Clustering using Frequent Itemsets. In Proceedings of StreamKDD’10, July 25, 2010, Washington, DC, USA.

[16] Un Yong Nahm, Raymond J Mooney, “Text mining with information extraction”, ACM, pp. 218, 2004.

[17] Xiangwei Liu, Pilian, “A Study On Text Clustering Algorithms Based On Frequent Term Sets”, Advanced Data Mining and Applications, Lecture Notes in Computer Science, 2005, Vol. 3584/2005, pp. 347-354, DOI: 10.1007/11527503_42.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] Zhou Chong, Lu Yansheng, Zou Lei, Hu Rong, "FICW: Frequent Itemset Based Text Clustering with Window Constraint", Vol. 11, n 5, pp. 1345-1351, 2006, DOI: 10, 1007/BF02829264.

[23] Zhitong Su, Wei Song, Manshan Lin, Jinhong Li, "Web Text Clustering For Personalized E-Learning Based On Maximal Frequent Itemsets", International Conference on Computer Science and Software Engineering, Dec 2008.

Author's Profile

Dr.M.Usha Rani is an Associate Professor in the Department of Computer Science and HOD for MCA, Sri Padmavati Mahila Viswavidyalayam(SPMVV Womens' University), Tirupati. She did her Ph.D. in Computer Science in the area of Artificial Intelligence and Expert Systems. She is in teaching since 1992. She presented many papers at National and Internal Conferences and published articles in national & international journals. She also written 4 books like Data Mining - Applications: Opportunities and Challenges, Superficial Overview of Data Mining Tools, Data Warehousing & Data Mining and Intelligent Systems & Communications. She is guiding M.Phil. and Ph.D. in the areas like Artificial Intelligence, DataWarehousing and Data Mining, Computer Networks and Network Security etc.