Maximal Frequent Itemsets Based Hierarchical Strategy for Document Clustering

Noor Asmat[1], Saif Ur Rehman[2], Jawad Ashraf[3] and Asad Habib[3]

Abstract— Text document clustering is an important issue in the field of information retrieval and web mining. Huge amount of text documents are needed to be clustered so that search engines can retrieve these documents efficiently and effectively. In this paper we present a novel approach for clustering text documents based on Maximal Frequent Item-set (MFI). Our proposed MFI-HC (Maximal Frequent Item-Set - Hierarchical Clustering) algorithm uses minimum k size MFIs to form initial sub-clusters. We use agglomerative hierarchical approach to merge the sub-clusters on the basis of highest similarity. Our proposed technique helps in reducing dimensions of the text and provides better clustering quality.

Keywords— Clustering, Hierarchical, Maximal Frequent Item-set.

I. Introduction

DUE to large corpus of text documents, we are normally facing problem of scalability in information retrieval. Besides scalability, the high dimensional nature of text documents create problem in clustering process as well. To retrieve the clusters easily, we need to tag them with relevant cluster labels, which is not possible in the case of high dimensional data. All these mentioned problems lead to the poor performance of information retrieval systems both in term of time and precision.

Today's web search engines have low precision as far as text document searching is concerned. When a user wants to retrieve some information, many irrelevant text documents are also retrieved. The challenges to confront include: retrieval of most relevant text documents, reducing high dimensionality of text documents and providing comprehensible description of text documents.

There is a room for improvement in current text document clustering system. Addressing the above mentioned problems, we present a novel approach to provide better quality clusters. Clustering is a technique to automatically group similar items. Clustering aims at lower inter-cluster similarity and higher intra-cluster similarity. I.e. all the clusters should be different from on another while the items of a single cluster should be similar. Frequent Item-set (FI) is a technique which is used to retrieve some frequent combination of terms present in text documents. Apriori is a well known FI extracting algorithm. Starting from 1 size Item-sets, Apriori works in k-steps. At each step k, it generates k-size Item-sets until the frequency of Item-sets at a particular step is lower than the threshold value [1]. FP-Growth is another FI extracting algorithm. FP-Growth works in a divide-conquer approach using FP-tree. The efficiency of FP-Growth is relatively very fast as compare to Apriori because FP-Growth reduces the size of dataset to be searched [2]. An FI will be known as MFI if there exists no super Item-set of that particular FI [3]. MFI is the compressed form of FIs and is the basis of our algorithm.

Partitioning, graph-based and Frequent Pattern based clustering are some of the techniques used for document clustering.

In partitioning method, the algorithm divides the dataset into desired number of non-overlapping groups. Each item is assigned to only one group. Some of the partitioning algorithms are k-means [4], variants of k-means like k-medians [5] and CLARANS [6]. Both k-means and CLARAN groups the data items based on higher similarity but k-means tries to minimize sum of squared differences while CLARAN tries to minimize cost of node with their respective medoids.

Graphical approach constructs graph containing text documents as vertices. Every two vertices are connected with edges and edges are assigned weights. Edges' weights are used to calculate the similarity between two documents [7]. Most of the graphical approaches use Hierarchical or tree based methods. Hierarchical method may either be agglomerative or divisive [8]. ROCK [9] and CHAMELEON [10] are the most popular hierarchical clustering algorithms. ROCK is an agglomerative hierarchical clustering algorithm. ROCK constructs a sparse graph of the data and then merges the clusters on the basis of aggregate inter-connectivity. CHAMELEON is another agglomerative clustering algorithm. CHAMELEON first divides the data into sub-clusters and then merges the sub-clusters on the basis of Relative Inter-Connectivity (RI) and Relative Closeness (RC).

Another approach is Frequent Pattern based clustering. In this approach clusters are created using FI extracted from any suitable FI discovery algorithm. Each cluster is assigned an FI as cluster label and all the supporting documents are assigned to that particular cluster. Frequent Pattern based clustering helps in reducing dimensionality of text documents and it provides understandable cluster label as well. FTC [11], HFTC [11], FIHC [12] and MFTCS [13] are some of the frequent item-set based text documents clustering algorithm. The detailed explanation of these algorithms is given in the section of related work.

The outline of our paper is as following. Section 2 of the paper briefly explains some well known related algorithms in detail. Section 3 presents our proposed framework, proposed algorithm and a solved example. Section 4 presents our experimental evaluation.

IIT, Kohat University of Science and Technology, Kohat-26000, Khyber, Pakhtunkhwa, Pakistan. Email: noor_kust28@yahoo.com

II. Related Work

Many researchers have presented different algorithms in literature. Hierarchical and Partitioning methods are the most commonly used approaches. Normally partitioning algorithms are based on some variants of Vector Space Model (VSM). VSM is based on words frequency and thus it is extremely affected by the high dimensionality of the text documents. These algorithms can not deal with high dimensional text documents especially in very large databases. To avoid this problem of high dimensionality of partitioning algorithms; many researchers used hierarchical approach in combination with Frequent Pattern based techniques. Frequent-Pattern-based algorithms help in reducing dimensionality of the text and are very efficient as compare to partitioning and hierarchical algorithms. Some of these algorithms are discussed here.

F. Beil et al presented two algorithms named as Frequent Term Clustering (FTC) and Hierarchical Frequent Term Clustering (HFTC). FTC is a flat clustering algorithm while HFTC is a hierarchical clustering algorithm. For the creation of clusters they used Apriori algorithm. FTC and HFTC used a greedy search approach to merge the cluster with maximum overlap. One of the draw back of FTC algorithm is that it only considers overlap of clusters. Moreover the greedy search approach can not guarantee optimum clustering system. [11].
FIHC [Frequent Item-Set Hierarchical Clustering] works in four phases. In first phase, initial Clusters are created using FIs extracted using Apriori algorithm. In second phase clusters are made disjoint by assigning the documents to the best cluster using a similarity score. In third and forth phases Parent-Child pruning and sibling merging are. FIHC is very slow for a small threshold value, and also small threshold value is required for higher precision. It is a big problem to balance between scalability and precision [12].

C. Su et al presented MFTSC (Maximal Frequent Term Set Clustering) algorithm. MFTCS extracts MFIs using FP-Growth [2] algorithm and then it merges these MFIs on the basis of a k-mismatch function. K-mismatch means that MFTCS will merges the cluster if the number of mismatch items between the clusters are less than k. K is a user specified integer value. To decrease the redundancy, clusters are merged based on an overlap function. After this some documents remain uncovered. Uncovered documents are assigned to the cluster of highest similarity with that particular document. The value K value may results in best or worst result depending on the nature of dataset. Moreover MFTSC is failed to provide disjoint clusters. The k-mismatch function is also a form of overlap so MFTCS uses overlap on different stages without considering any other issue [13].

All the algorithms were good attempts but still there are some common problems that remain unsolved. For higher support values, these algorithms return low quality clusters but improved run time while low support value returns good quality cluster but takes a lot of time for larger datasets.

For low support, FIs discovery algorithms return a very large number of small size FIs or MFIs that ultimately leads to the slow performance of algorithm and increases the run time at a very high rate. Reason behind the slow run time is that larger amount of FIs or MFIs need a larger amount of merging cycles. Thus we need to reduce the number of MFIs to get optimum results.

Another problem is that most often these small sizes FI or MFIs take several cycles to merge smaller FIs or MFIs to their final MFIs while merging sub-clusters to the most similar clusters again and again. We need to merge each sub-cluster to its most final destination without passing through unnecessary cycles.

III. Proposed Frame Work

The algorithm we propose is based on an agglomerative hierarchical approach that helps in reducing dimensionality and improving efficiency. Our algorithm helps to avoid repeating unnecessary merging cycles as it eliminates small size MFIs and quickly merges sub-clusters to their final clusters. Small size MFIs are eliminated by giving an extra parameter to the FP-Growth algorithm that is the minimum size of MFI. Produced MFIs are used for the creation of sub-clusters and the labels of that particular cluster. The use of MFIs instead of FIs improves effectiveness and accuracy at an amazingly rate and the elimination of small size MFIs adds further improvement. The main phases of the proposed algorithm can be expressed by the following diagram.
;

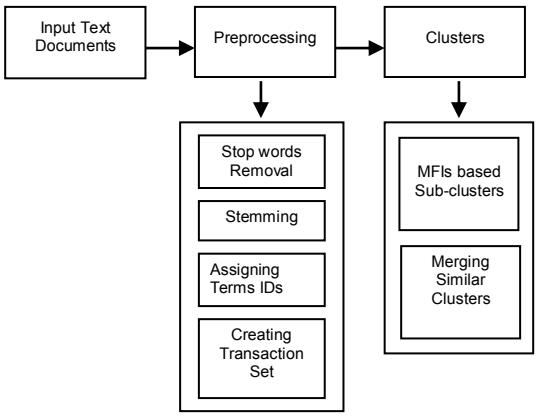


Fig. 1. System Architecture

To implement our algorithm, we have used Classic4, Reuter8 and Wap datasets downloaded from [14]. Preprocessing steps include stop-words removing and stemming. Moreover terms are replaced with their respective terms IDs and documents are converted to transactional form. After performing these necessary preprocessing steps, MFIs based sub-clusters are created and then similar sub-clusters are merged to create final clusters.

Algorithm:
Procedure MFI-HC ( )
Input: D, *k*, Min_Sup.
Output: Clusters
Method:
1. MFI=FP-Growth(D, k, Min_Sup);
2. C=Assign_Docs(D_k, MFI);
3. for n Num_Of_Clusters
   for each Cluster C_i € C
   Loc_Max_Sim= Sim(C_i, G);
   Merge(C_i, $C_{Loc\_Max\_Sim}$);

Remove C_i, $C_{Loc\_Max\_Sim}$;
end
end
4. return Cluster;

D is a set of transactions that we have already created. FP-Growth is an MFI discovery algorithm that takes a set of transaction as input. MFIs of min-size *k* are extracted for a specific support value. Based on these extracted MFIs, sub-clusters are created and the terms present in MFIs are considered as the labels of the clusters. For the creation of initial sub-clusters and for merging of similar clusters, we need a scoring function for the purpose of identifying the best cluster. The scoring function we used is based on the overlap between two patterns. The overlap may exist between two cluster points or between document and a cluster point. Our scoring function is basically the sum of intersection of two sets and the percentage of the size of set for which we are going to compute the merging score. The purpose of adding size factor is because we want to merge the cluster with the most similar and largest cluster to avoid extra cycles of merging.

The following scoring function is used for finding the best match for a document.

$$Score(C \leftarrow D_i) = Max\ C_j\ \in C\ \{\frac{|D_i\ \cap C_j| + |C_j|}{100}\}$$

In the above function, D_i represents a document *i* and C_j represents a sub-cluster *j*. Here $|\ D_i\ \cap\ C_j|$ represents the cardinality of the intersection between D_i , C_j and $|C_j\ |$ represents the cardinality of set j. The score of D_i will be calculated against a number of sub-clusters say C_j where C_j € C. The function allots highest score to the sub-clusters that have highest overlap and is the largest in size. The same procedure will be performed for merging of clusters, but instead of comparing documents D_i with clusters, cluster to cluster comparison will be performed. For merging of cluster we will use the above scoring function with a slight change.

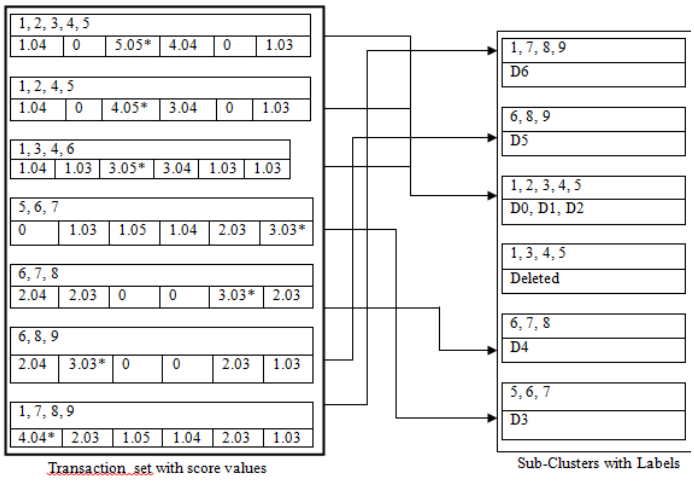


Fig. 2 Creating Disjoint Sub-Clusters

$$Score(C \leftarrow C_i) = Max\ C_j\ \in C\ \{\frac{|C_i \cap C_j| + |C_j|}{100}\}$$

Suppose we have a document-set consisting of 7 documents. After performing necessary preprocessing steps, we got the following transaction-set and their respective MFIs.

TABLE I
TRANSACTION_SET

Doc_ID	Term_ID	MFI_ID	MFIs at min-size= 0, 1,2,3 Terms_ID	MFIs at min-size=4 Terms_ID
Doc_0	1, 2, 3, 4, 5	M0	1, 7, 8, 9	1, 7, 8, 9
Doc_1	1, 2, 4, 5	M1	6, 8, 9	1, 2, 3, 4, 5
Doc_2	1, 3, 4, 6	M2	1, 2, 3, 4, 5	1, 3, 4, 5
Doc_3	5, 6, 7	M3	1, 3, 4, 5	
Doc_4	6, 7, 8	M4	6, 7, 8	
Doc_5	6, 8, 9	M5	5, 6, 7	
Doc_6	1, 7, 8, 9			

For min-size=4, FP-Growth algorithm will delete all the MFIs of size 1, 2 and 3. For above set of transactions min-size=0, 1, 2, 3 returns the same set of MFIs because the above transactions have no MFI of size 1 or 2. To explain the algorithm, we select min-size=3 and the complete procedure of assigning the documents to the relevant clusters is shown in Fig 2.

The nodes at the left side of the graph represent documents and the nodes at the right side of the graph represent sub-clusters. For each document, the scoring values are calculated against each sub-cluster and then the document is assigned to the cluster of highest score value. The scoring value of Doc_0 for all the sub-clusters are computed and are given in the first node of the graph as below. To assign the documents D0 to the most suitable sub-cluster, the similarity score is calculated as following.

$$Score(M_0 \leftarrow D_0) = \frac{|\ Intersection\ (D_0,\ M_0)| + |M_0|}{100}\}$$

= 1+4/100
=1.04

Here Score(M0← D0) represents the score of assigning document_0 to MFI_0, D0 represents the first document and M0 represents the first MFI. | Intersection (D0, M1) | represents the number of items that are common to both D0 and M0. |M0| represents the number of items in M0. Here 1.04 is the calculated Score(D0←M1), 0 is Score(D0←M1) and 5.05 is Score(D0←M2) etc. D0 will be merged with sub-cluster M2 as D0 has highest score for M2. Above procedure is repeated until all the documents are assigned to any of the sub-cluster and we get the following sub-clusters.

TABLE II
ASSIGNED DOCUMENTS

MFI_Set	Terms_ID	Documents
M0	1, 7, 8, 9	D6
M1	6, 8, 9	D5
M2	1, 2, 3, 4, 5	D0, D1, D2
M3	1, 3, 4, 5	Deleted
M4	6, 7, 8	D4
M5	5, 6, 7	D3

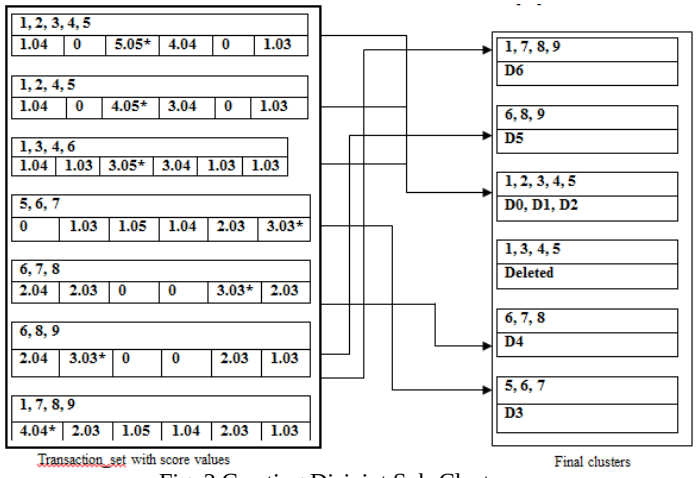


Fig. 2 Creating Disjoint Sub-Clusters

Nodes at the left side of the above graph represents sub-clusters and the nodes at right side represents final clusters that are been produced after merging. To merge the sub-cluster M0, the scores of M0 with all the rest of the MFIs are calculated and are presented in the first node of the graph at left.

$$Score(M_0 \leftarrow M_1) = \frac{|Intersection\ (M_0,\ M_1)| + |M_1|}{100}\}$$

= 2+3/100= 2.03

Sub-cluster M0 has highest scoring value for M1, so M0 is merged with M1. Thus we get two final clusters that are been shown at the right side of the graph with the documents and their respective cluster labels. We used an agglomerative hierarchical clustering approach for the clustering and the overall clustering process using agglomerative hierarchical approach is given.

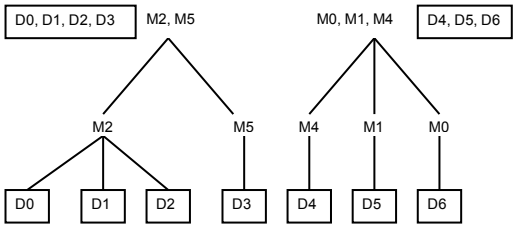


Fig. 4 Agglomerative Hierarchical Approach

IV. Results And Evaluation

In this section we evaluate the performance of our proposed framework based on different min-size of MFIs. For the same dataset we executed FIHC based on similar support values. FIHC is based on hierarchical approach and also using FIs for clustering. We executed our program on Window 7 with core 2 duo 1.33 GHz CPU and 2 GB memory. We obtained the FIHC 1.0 from the [15].

F-measure is the most commonly used evaluation measure. Good clustering algorithm aims to capture larger number of documents from a natural class. F-Measure is based on recall and precision [12]. Precision is the term representing all correctly captured documents while recall is the term representing all the captured documents regardless of accuracy. The recall, precision and F-Measure will be calculated as follows:

$$Recall(|K_i, C_j|) = \frac{n_{ij}}{|K_j|}$$

$$Precision(|K_i, C_j|) = \frac{n_{ij}}{|C_j|}$$

$$F(K_i, C_j) = \frac{2 * Recall(|K_i, C_j|) * Precicion(|K_i, C_j|)}{Recall(|K_i, C_j|) + Precicion(|K_i, C_j|)}$$

Where K is the natural class of dataset and j is the cluster. n_{ij} is the number of documents of class K_i that are present in cluster C_j. |Ki| is the number of documents in class i and $|C_j|$ is the number of documents in cluster j. The weighted sum of maximum F values for all the natural classes is known as F of clustering result.

$$F(C) = \sum_{k \in K} \frac{|k_j|}{|D|} max_{C_{j \in C}} \{F(K_i, C_j)\}$$

We used the most commonly used datasets e.g. Classic4, Reuters8 and Wap. All the used datasets are collection of English text documents and contains different number of natural clusters. The following table shows a short summary of these datasets.

TABLE III
Dataset

Dataset	# Docs	# of Classes
Wap	3900	21
Classic	7095	4
Reuter	7674	8

The F-Measure and run time comparison of the above datasets based on different parameters are shown in detail. For MFI-HC we used min-size=1, 2, 3 and 4 and for FIHC we used 30% and 40% support. The number of clusters for both algorithms remains same. The produced results are shown below.

TABLE IV
F-MEASURE AND TIME OF CLASSIC4 DATASET

Cluster	Sup	F-Measure						Time					
		MFIHC				FIHC		MFIHC				FIHC	
		1	2	3	4	30%	70%	1	2	3	4	30%	70%
5	1	0.44	0.53	0.54	0.45	0.59	0.44	452	325	93	12	304	305
	2	0.44	0.46	0.27	0.45	0.52	0.42	70	35	2	1	64	58
	3	0.55	0.23	0.0	0.45	0.60	0.53	24	9	1	0	21	21
10	1	0.56	0.50	0.56	0.45	0.55	0.41	432	334	94	12	335	285
	2	0.44	0.42	0.24	0.45	0.49	0.40	69	35	2	1	69	66
	3	0.59	0.23	0.0	0.45	0.58	0.48	24	9	1	0.5	24	21
20	1	0.45	0.51	0.53	0.45	0.51	0.41	447	321	89	12	319	273
	2	0.44	0.42	0.24	0.45	0.46	0.40	69	35	2	1	78	64
	3	0.53	0.23	0.0	0.45	0.46	0.40	24	9	1	0.5	63	56
40	1	0.47	0.47	0.52	0.45	0.47	0.38	321	90	89	12	352	315
	2	0.44	0.42	0.24	0.45	0.45	0.37	69	35	2	1	71	58
	3	0.53	0.23	0.0	0.51	0.42	0.36	21	9	1	0.5	21	6

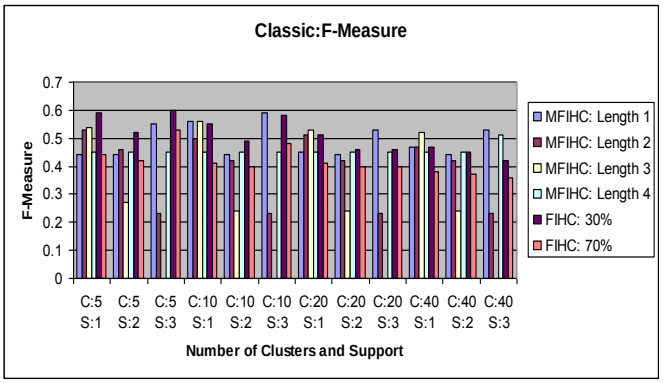
TABLE V
F-MEASURE AND TIME OF REUTER8 DATASET

Cluster	Sup	F-Measure						Time					
		MFIHC				FIHC		MFIHC				FIHC	
		1	2	3	4	30%	70%	1	2	3	4	30%	70%
5	1	0.51	0.51	0.50	0.49	0.70	0.54	1090	2359	2187	1784	1075	2419
	2	0.50	0.50	0.62	0.49	0.66	0.67	368	367	336	244	163	151
	3	0.49	0.58	0.61	0.51	0.71	0.66	128	127	106	67	62	60
10	1	0.51	0.50	0.57	0.49	0.65	0.54	2377	2396	2213	1774	997	974
	2	0.50	0.45	0.61	0.49	0.68	0.70	369	364	325	219	153	150
	3	0.48	0.54	0.61	0.51	0.69	0.53	128	127	106	66	62	59
20	1	0.51	0.51	0.69	0.49	0.65	0.57	2366	2410	2266	1842	998	972
	2	0.49	0.45	0.61	0.60	0.68	0.63	369	364	323	219	153	153
	3	0.60	0.64	0.56	0.51	0.69	0.52	128	127	104	65	62	59
40	1	0.51	0.49	0.70	0.47	0.65	0.60	2358	2422	2247	1791	994	970
	2	0.47	0.49	0.65	0.60	0.68	0.68	367	360	320	217	153	147
	3	0.60	0.65	0.50	0.29	0.65	0.52	127	131	109	67	63	59

TABLE VI
F-MEASURE AND TIME OF WAP DATASET

Cluster	Sup	F-Measure						Time					
		MFIHC				FIHC		MFIHC				FIHC	
		1	2	3	4	30%	70%	1	2	3	4	30%	70%
5	1	0.38	0.38	0.38	0.29	X	X	175	165	164	165	x	x
	2	0.38	0.38	0.38	0.29	X	X	34	32	32	32	x	x
	3	0.38	0.38	0.38	0.29	0.44	0.36	12	11	12	12	390	370
10	1	0.38	0.38	0.38	0.29	x	x	175	166	168	165	x	x
	2	0.38	0.38	0.38	0.29	x	x	33	31	31	31	x	x
	3	0.38	0.38	0.38	0.38	0.53	0.43	12	12	11	12	394	397
20	1	0.38	0.38	0.38	0.29	x	x	174	172	167	166	x	x
	2	0.38	0.38	0.38	0.29	X	x	34	33	32	32	x	x
	3	0.38	0.38	0.38	0.38	0.56	0.43	12	12	12	11	425	374
40	1	0.38	0.38	0.38	0.29	x	x	172	166	161	161	x	x
	2	0.38	0.38	0.38	0.29	x	x	33	31	32	31	x	x
	3	0.38	0.38	0.38	0.29	0.50	0.42	12	11	11	11	410	392

The above results are expressed graphically to give a clear picture.



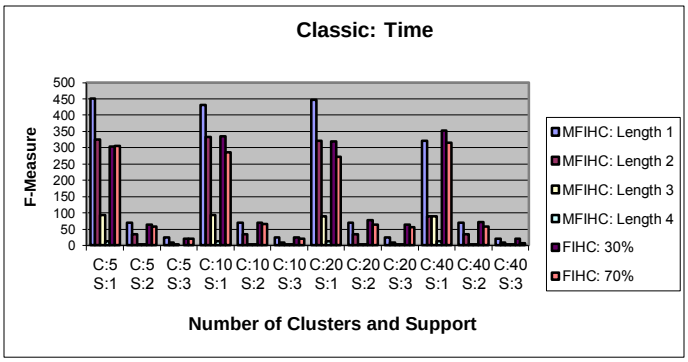
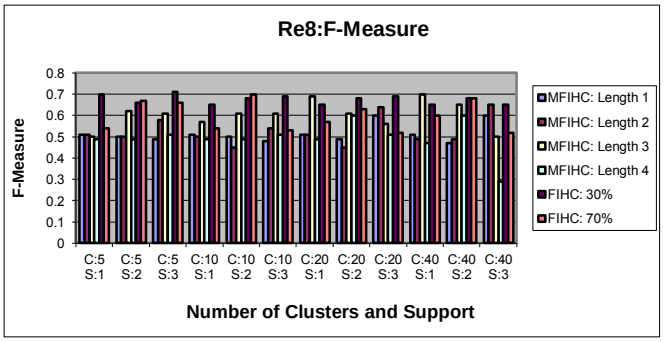


Fig. 5. F-Measures and Time of classic4 dataset



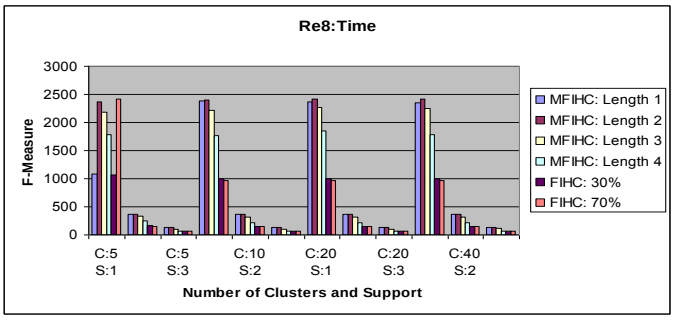
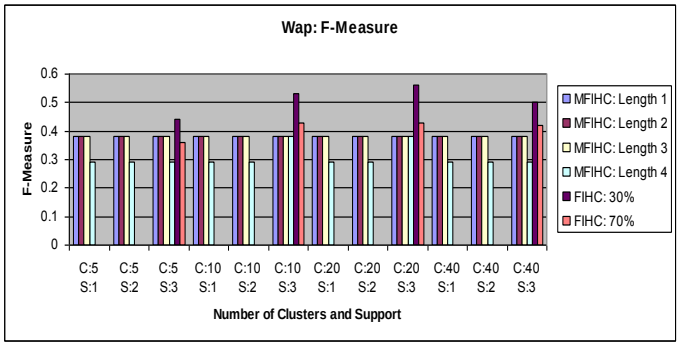


Fig. 6. F-Measures and Time of Reuter8 dataset



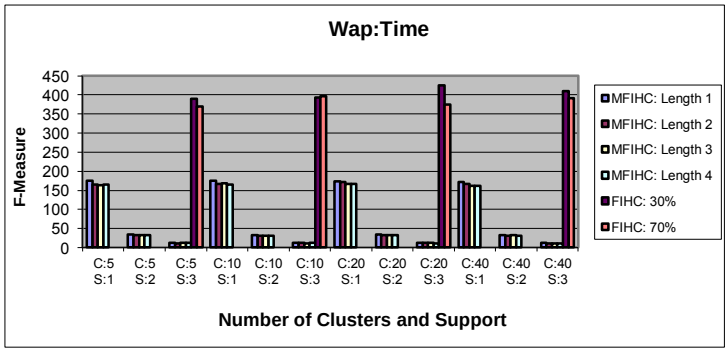


Fig. 7 F-Measures and Time of Reuter8 dataset

From the above results we can easily realize that MFI-HC is more efficient as compare to FIHC. For classic and wap MFI-HC have a prominent improvement in run time of the clustering algorithms. For Reuter8, FIHC is more efficient than FIHC but it can give efficient result if we increase the support. For wap dataset both in MFI-HC and FIHC, if we select a very small support then it takes very huge amount of memory, slows down the system and is unable to perform clustering.

For the cluster quality i.e. F-Measure, FIHC has different trends over different parameters. FIHC produce good F-Measure for smaller cluster support but is not efficient, while larger cluster support can produce good F-Measure but is not very efficient. From the results of classic dataset we can see that for 70% cluster support FIHC has a better clustering quality but is not efficient and for 30% cluster support MFI-HC outperforms FIHC both in terms of run time and F-Measure. For the above set of parameters MFI-HC could not produce good F-Measure while FIHC has produced relatively good clustering quality. From the F-Measures produced by MFI-HC, we can see that for some set of parameters we get very small F-Measure i.e. below 0.30, the reason behind this decrease of F-Measure is that the number of required clusters is less than the number of produced MFI's. Thus algorithm fails to merge similar clusters.

In short, we can say that using different parameters, both MFI-HC and FIHC has different impact on run time and quality. For MFI-HC if we select larger size of MFIs then the algorithm produce good quality both in terms of rum time and cluster quality. And FIHC can produce good quality for very lower support but is very slow.

The nature of dataset is no doubt a very important factor for any clustering algorithm. No single algorithm can produce perfect results for all types for environment and same is the case with MFI-HC.

V. Conclusion

Most of the existing text clustering approaches does not assure the needs of clustering system. They are not capable of

dealing with large volume of data, high dimensionality and cluster labels. Sometime they can not produce required number of clusters. MFI-HC has improved scalability and improved dimensionality reduction technique. MFI-HC can produce good clustering label as well. Our experimental evaluation shows that MFI-HC outperforms the existing clustering techniques both in terms of efficiency and clustering labels.

REFERENCES

[1]. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proceedings of the 20th International Conference on "Very Large Data Bases, VLDB", pages 487-499, Santiago, Chile (1994)

[2].Borgelt, C.: An Implementation of the FP-growth Algorithm. In: Workshop "Open Source Data Mining Software (OSDM'05, Chicago, IL)", pp. 1-5, ACM Press, New York, NY, USA (2005)

[3]. M., Burdick, Calimlim, M., Gehrke, J.: MAFIA: A Maximal Frequent Itemset Algorithm for Transactional Databases. In: Proceedings of the 17th BIBLIOGRAPHY 63 International Conference on "Data Engineering", pages 443-452, Heidelberg, Germany (2001)

[4]. Hartigan, J. A., Wong, M. A.: Algorithm AS 136: A K-Means Clustering Algorithm. In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, Vol. 28, pp. 100-108, Royal Statistical Society (1979)

[5]. Bradley, P. S., Mangasarian, O. L., Street, W. N.: Clustering via Concave Minimization. In : *Journal of Advances in Neural Information Processing Systems*, Vol. 9, pp.368-374,MIT Press (1997)

[6]. Ng, R., Han, J.: Efficient and effective clustering method for spatial data mining. In: Proc. of the 20th Conference on "VLDB", pages 144–155, Santiago, Chile (1994)

[7]. AbdelHamid, N. M., Halim, M. B. A., Fakhr, M. W.: Bees Algorithm-Based Document Clustering. In: The 6th International Conference on "Information Technology", College of Computing and Information Technology, Cairo, Egypt, 246-254, (2013)

[8].Sasirekha, K., Baby, P.: Agglomerative Hierarchical Clustering Algorithm- A Review Similarity Measures. In: *International Journal of Scientific and Research Publications*, 3, pp. 1515-1518, (2013)

[9]. Guha, S., Rastogi, R., Shim, K.: ROCK: A Robust Clustering Algorithm for Categorical Attributes. In: Proceeding of 15th International Conference on "Data Engineering", IEEE CS Press, Los Alamitos, Calif, pp. 512-521, (1999)

[10] Karypis, G., Han, E., Kumar, V.: Chameleon: Hierarchical Clustering Using Dynamic Modeling. In: *IEEE Computer, Special Issue on Data Analysis and Mining*, vol.32, pp. 68-75, (1999)

[11] Beil, F., Ester, M., Xu, X.: Frequent Term-Based Text Clustering. In: Proceedings of the eighth ACM SIGKDD international conference ;on "Knowledge discovery and data mining, ACM, New York , 436, (2002)

[12]. Fung, B., Wang, K., Ester, M.: Hierarchical Document Clustering Using Frequent Itemsets. In: Proceeding of SIAM International Conference on "Data Mining", SIAM, pp.59-70, (2003)

[13]. Su, C., Chen, Q., Wang, X., Meng, X.: Text Clustering Approach Based On Maximal Frequent Term Sets. In: Proceeding of 2003 IEEE International Conference on "Systems, Man and Cybernetics", Harbin Institute of Technology, Shenzhen, China, pp.1551-1556, (2009)

[14] Index of /torch/datasets, http://sites.labic.icmc.usp.br/torch/datasets/,(18 Nov, 2014)

[15].FIHC 1.0, http://ddm.cs.sfu.ca/, (18 Nov, 2014)