CO461- Data Warehousing Data Mining

A REPORT ON THE PROJECT ENTITLED

Text Clustering using Priority based Maximum Capturing

Submitted by

JELWIN RODRIGUES **RAVIKIRAN RUDRAPPA TALAWAR**

171CO218 171CO234

Department of Computer Science and Engineering
National Institute of Technology Karnataka, Surathkal

December 2020

Abstract

We get frequent itemsets using association rule mining. By generating frequent itemsets from various documents, it is possible to classify these documents into various clusters based on some defined algorithms. This is done by applying various text mining algorithms on the collected documents. In this project, we have implemented the improvised version of the Maximum Capturing algorithm for text clustering. This clustering approach is mainly based on the similarity index between two words. We can calculate the similarity index between 2 words by many different existing algorithms. Some of these methods are Distance calculation between the vector representation of the words, Cosine measure to find the similarity, Hamilton distance, Euclidean distance etc. The Levenshtein distance between two words is the minimum number of single-character edits required to completely change a word into another word. Levenshtein distance is used for calculating similarity index in our case.

Once we get the similarity index between the documents in the form of a matrix representation, we can use various algorithms to cluster the raw word data into various clusters, where the final normalized cluster size must be equal to the number of label types present in the dataset. Some various cluster algorithms that exist currently are Clustering based on Frequent Word Sequence (CFWS), Document Clustering Based on Maximal Frequent Sequences (CMS), Frequent Term-Based Clustering (FTC), Frequent Itemset-based Hierarchical Clustering (FIHC) and Maximum Clustering method (MC). Researches done have shown that the Maximum Clustering method performs best when compared to all of the other clustering algorithms. There are 3 different algorithms that are present in Maximum Clustering method, first one is to get all the clusters, then next one gives us frequent itemsets in each of these clusters, and last one is to normalize the clusters so that we have only a particular number of clusters. Finally we have suggested an improvement for the existing MC algorithm, where we have assigned priorities for the documents having equal similarity values whenever there is a need to obtain maximum similarity value document pair and the max value is equal.

Table of contents:

I. Introduction:

(a) Brief Statement of the problem:

Title: Text Clustering using Priority based Maximum Capturing

The main goal is to construct document clusters and we try assigning these clusters based on frequent itemsets. To produce these clusters, we intend to use frequent items and try to represent the clusters based on these frequent items, and also another step in getting clusters is to calculate the similarities between the texts and store these similarity values in a similarity_index matrix. The algorithm that is used to construct the document clusters for our problem is minimum spanning algorithm, that is we start with the minimum distance value and keep expanding the cluster based on similarity index matrix. The approach used is Maximum clustering .

We take a document D is denoted by the count of itemsets in it.

That is, $Doc_j = \{F_{j1}, \ldots, F_{jn}\}$, where n represents all the itemsets present in the document. We now define the similarities between the 2 items as the number of frequent itemsets they have in common. We define a similarity matrix M, with M[i][j]=similarity[i][j] if i<j, else M[i][j]=0. Using these values and the Maximum clustering algorithm which will be explained later in detail, the itemsets are clustered into various clusters, then normalized to reduce the number of clusters equal to the number of class labels that we have used for our dataset, and then frequent items are extracted from each cluster to get the desired output for our problem.

(b) Importance/novelty of the problem:

The motivation of defining this above problem lies in adopting frequent itemsets to help in document clustering. The purpose of using these frequent itemsets is by finding frequent itemsets we get how a single word is conceptually as well as contextually important for clustering. There has been a lot of research which has been done in the field of Text clustering as well as Frequent itemsets generation. The existing algorithms for the same text clustering are Clustering based on Frequent Word Sequence (CFWS), Document Clustering Based on Maximal Frequent Sequences (CMS), Frequent Term-Based Clustering (FTC), Frequent Itemset-based Hierarchical Clustering (FIHC) and Maximum CLustering method (MC) . We have proposed an additional step to the current MC algorithm that exists. This will help us increase the accuracy of the dataset that we use.

The key details that were explored during research of MC algorithm was we initially construct the similarity matrix M, find the minimum and maximum values in this matrix and based on the 2 maximum values, clustering is done. But, in the current methodology that is proposed by researchers, there is no mention of what needs to be done when the similarity matrix values of 2 pairs of texts equal. So suppose there are 2 text pairs, (a,b) and (c,d) and they have their similarity matrix value equal, how is the priority given in this case? This must be taken into care so that the algorithm performs better when it encounters data of this form

and we have proposed a new improvement to the existing maximum clustering algorithm by defining the algorithm for equal similarity case.
So, the extra component that we have added to the existing maximum clustering algorithm is that whenever the similarity values between 2 pairs of text are same, higher priority is given to the text which is not already clustered before. So the priority order is " No text clustered among the 2" then "one of the text clustered before, then "both the text clustered", which cannot be clustered further as it is already clustered. For example say we have 2 text pairs to be clustered, (a,b) and (c,d), Let us assume only 'a' is clustered already. Now since one of the text is clustered in (a,b) and none of the text is clustered in (c,d), higher priority is given to the (c,d) cluster as the rules defined and (c,d) is clustered before (a,b).

(c) Related literature:

In this section we provide some of the literature review that are done based on our problem statement so that we can get to know the existing research areas and find out the research gaps for implementation and improvement of our problem statement. The Maximum clustering proposed can be divided into two components: constructing document clusters and assigning cluster topics.[1] The algorithm used to construct the document clusters is the minimum spanning algorithm,in which initially we have a node and one by one every document is added this node based on the similarity index values calculated.[1] The basic idea of MC is that if say suppose we have a document pair having a very high similarity index value, then both the documents that belong to this pair must be clustered before other document pairs.[1] MC produces the clusters by maximizing the similarity values between the documents. If say both the documents do not belong to any clusters it prefers to add a new cluster other than adding these documents to the existing clusters.[1] After al the clusters are produced there is a need to normalize these clusters so that we will have only certain number of clusters equal to the number of types of class labels present in the dataset. Reuters-21578 document collection was the dataset used for implementation of MC algorithm.[1] Finally, afte preprocessing we get a total of 6 categories. Stop word analysis and stemming process are conducted as a part of the preprocessing task.[1] FP-tree algorithm is used to extract frequent itemsets from document. F-measure is employed to estimate performances of the Maximum clustering method.[1]

A global frequent itemset is defined as the set of items that are present together whose value is greater than the minimum fractions present in the whole document.[2] The set of clusters that we have received earlier is now helpful for construction of a tree with the clusters can be viewed as a set of topics present in the whole document.[2] The final output tree that we get has 2 main objectives one is to form a foundation for pruning and the other to found a foundation for browsing the data. In this tree, every cluster has exactly one parent. The topic of a parent cluster has more value than topic of the child cluster and we can say that these 2 cluster values are highly similar.[2] The main goal of clustering here is to make a group of similar points in a feature space which is mainly based on proximity, so that this way we get a hierarchical structure.[3] We get this sort of results using unified time complexity technique from k-means and scatter graph. Using various partitional algorithms , we get hierarchical structures recursively.[3] The overall time complexity of the above algorithm suppose we have N documents is O(N log N). We can conclude that using weighting extracting terms from TD-IDF helps us produce much better results than only using unit frequency for all the documents. Also, we can show that truncating feature vectors saves time as we can prove that length of the vector and time neede dto cluster are fairly proportional. [3]

Results have proved that the bisecting K-means clustering has produced better results than the normal K-means method.[4] In bisecting K-means[4] , we initially pick a cluster to split. We then try to find 2 sub-clusters using the existing K-means algorithm. We then continue this process number of times, number which is defined by us and split this based on the higher similarity index value. Repeat all the process like its done in k-means.[4] Next, we move on to the FP-tree-based pattern-fragment growth mining method.[5] FP-tree[5] is generated by recursively performing various mining operations and using a tree to represent the entire process with the key value at every node.[5] The pattern growth is achieved by combining the suffix pattern with the new ones that are produced from a conditional FP-tree. Since we use only frequent itemsets play an important role in frequent pattern mining to obtain a FP tree, we initially need to scan through the DB to obtain the frequent itemsets in descending order of their count values.[5] We represent the items in the root of the FP tree as follows (a :4), (b:4), (c:3), (d:3), (e:3), (f:3) . Advantage of using FP tree over Apriori is that it ensures it never produces candidate set combinations of itemsets not present at all in the dataset, as the items present in the dataset is encoded in the FP tree path. [5]

(d) Scope of the project :

The current implementation of the project is capable of clustering a list of words into multiple clusters and list the frequent itemset for each newly formed cluster based on a defined minimum support value. The similarity of words is calculated by comparing the characters in the word. This implementation could be modified to accept a list of sentences and compare the words in it and calculate the similarity in the sentences, and the similarity matrix could be constructed accordingly. Thereafter making the clusters using this matrix. Similarly, this project could be used to cluster the documents either by considering the headings in it or the predefined phrases.

(e) Brief statement on subsequent chapters:

Up till now we have given the basic introduction to our problem statement along with the title and also come up with a new novelty to our project. Then we did the literature review of the related papers to get to know the research that is already been done in this field. Now in the coming topics we start with the methodology of our project. In section II, we explain the implementation of our project and all the steps and algorithms that we have implemented are written in detail . In section III, we publish the results that we have obtained during the implementation like various matrices received as output, accuracy value, final cluster after complete implementation along with the frequent itemsets etc. Finally in section IV, we end this report by concluding and highlighting the key points covered in this entire implementation.

II. Materials and Methods:

(a) Implementation:

We carry out the implementation of our project using Google Colab environment using an ipynb file. So, initially we need to import all the necessary packages that are needed so that we can use the needed functions during the implementation. The packages that we need are Numpy to covert the data in vector form, pandas is also used to read the csv file which is our raw dataset. One Other packages used is distance, to calculate the similarity distance between 2 document items, The dataset that we have used is Reuters-21578 document collection, which is available online. We have preprocessed this data and reduced the number of categories to 4 and number of tuples totally present are 156 (39 of each type). Once we have this raw CSV data , we convert this data to numeric form using numpy library and store the text section in one variable and the label is stored in another variable. Once this is done, now we have the words in an array, with which we start our algorithm to get the clusters. Totally we have 3 algorithms implemented. First one is to get all the clusters, then next one gives us frequent itemsets in each of these clusters, and last one is to normalize the clusters so that we have only a particular number of clusters which is equal to the number of label types that are present in the dataset.
Once all these algorithms are implemented, we get normalized clusters , whose number equals number of label types present in dataset. In our case we have taken the dataset with 4 label types, so finally after implementing all the algorithms which are explained in detail later, we get 4 clusters and now our task is to find out the accuracy in determining these clusters. We evaluate our project using F1 score and accuracy metric is used to get an accuracy of 69% on the training data. We also construct the tabular representation which shows the number of type values in every cluster, which is a 4*4 matrix.

(b) Algorithms:

There are 3 algorithms which we have implemented. Let us discuss each of these 3 in detail:
i) Algorithm to generate the clusters:

1. Initially we need to calculate the similarity matrix M. The similarity index between the 2 itemsets is calculated using levenshtein distance from the “distance” library imported , which gives us how close one word in the document pair is with respect to another word. Similarly if we do for lall the document pairs present, we get a similarity matrix M, of size N*N, if there are N documents.
2. Find the minimum value in similarity matrix M, excluding any zero present in this matrix.
3. Find the max value present in the similarity Matrix M. Once we find the maximum value what we get is 2 document clusters with this value in which at least one of these documents are unclustered before. If both the documents are clustered before, assign

the similarity Matric value for this pair equal to 0 and repeat this step to get the document pair with next maximum value.

4. If suppose there are 2 text pairs having their similarity matrix value equal, then we give priorities to these document pairs based on their clustering status. The priorities we have assigned are “ No text clustered among the 2” then “one of the text clustered before, then “both the text clustered”. This step is needed while the maximum value calculated from step 3 might be same for many document pairs and we need to assign which documents gets clustered first in this case.
5. Once we get the document pair to be clustered to the main cluster using Minimum spanning algorithm, if one of the documents is already clustered before , then we add the document which is not yet clustered to the main cluster which contains the other document from this pair. If say both are not clustered yet, then we make a new cluster and add both these documents to this cluster. In this step, if the similarity index is equal to minimum similarity index value, then both the documents in this pair are clustered separately. We repeat from Step 3 now.

(ii) Algorithm to get the frequent itemsets:

1. We select the most frequent itemsets with the length equal to the max length for every cluster in its candidate topic.
2. We do the 2 process in step 2 to conduct reduction of these candidate topics
    a. Assign the frequent itemsets with the maximum lengths calculated as topic for each cluster present.
    b. For every frequent itemsets calculated in step 1, if it is present in the topics list in step 2.1, then the frequent itemsets must be eliminated from the cluster’s candidate topics.

(iii) Algorithm to normalize the cluster:

1. The learning rates of all the clusters produced by Maximum Clustering algorithm must be initialized to 0, and also we append all the clusters to a set.
2. We find the learning rate of all the clusters that are present in the cluster set, and we find the maximum similarity value among these clusters.We terminate the process if we get the value as 0. Else, we randomly select one cluster, say cluster A from the set.
3. We find another cluster similarly, say cluster B which has maximum similarity with cluster A. If the value is again 0, we go back to step 2.
4. We merge the 2 clusters A and B. We also sum the learning rates of 2 clusters , cluster A and cluster B and assign it to a new cluster’s learning rate value.
5. We now remove the cluster A and cluster B from the cluster set and we add the new cluster into the cluster set.

(c)Evaluation:

After all the clusters are normalized into set number fo clusters, we need to evaluate the performance of our algorithm. We use F-1 accuracy to determine the accuracy of our project. Initially we get the matrix representation from the clusters that we have calculated.

Comparing, for every label type in the main dataset, we check for the corresponding number of types in each clusters present in the final normalized cluster. So, for every cluster we get 4 values, that are th number of items from every label type and 4 because we have 4 label types. Since we have 4 clusters finally we get a matrix of size 4*4. Let us represent this 4*4 matrix A as follows,

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

Now we calculate the accuracy using F1 metrics as follows;

$$\text{Accuracy}= \frac{a_{11}+a_{22}+a_{33}+a_{44}}{Sum\ of\ all\ elements\ in\ Matrix}$$

Using this we get an accuracy of 69% for our implementation and dataset.

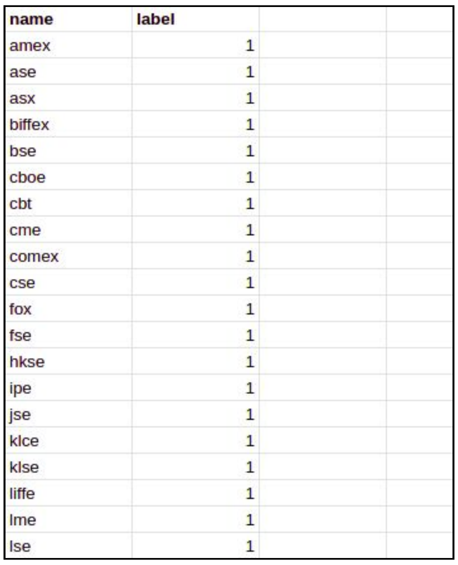
(d) Improvements/ New contributions:

The existing Maximum capturing algorithm does not take into account the case when the similarity values are equal while deciding the maximum value to get the document pair. So, the extra component that we have added to the existing maximum clustering algorithm is that whenever the similarity values between 2 pairs of text are same, higher priority is given to the text which is not already clustered before. So the priority order is " No text clustered among the 2" then "one of the text clustered before, then "both the text clustered", which cannot be clustered further as it is already clustered. For example say we have 2 text pairs to be clustered, (a,b) and (c,d), Let us assume only 'a' is clustered already. Now since one of the text is clustered in (a,b) and none of the text is clustered in (c,d), higher priority is given to the (c,d) cluster as the rules defined and (c,d) is clustered before (a,b).

Using this improvement to the existing algorithm ,we test the same on the current dataset which give sus an accuracy of 69%. Without this additional improvement on the same data we get an accuracy of 66.6%, which implies that the additional changes or the new contributions made on this topic is an improvement that we get for this current dataset.

III. Results and Discussion :

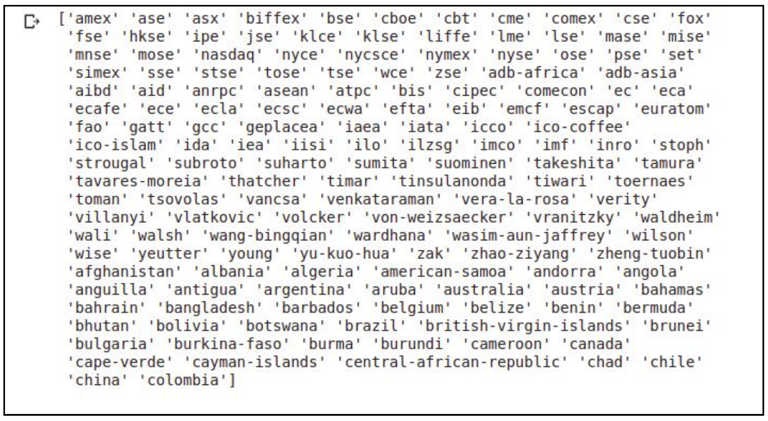
We have carried out our implementation using the Reuters-21578 document collection, which is available online as our dataset. We have preprocessed this data and reduced the number of categories to 4 and number of tuples totally present are 156 (39 of each type). The below figure , Fig 1 shows the dataset that we have used:



name	label		
amex	1		
ase	1		
asx	1		
biffex	1		
bse	1		
cboe	1		
cbt	1		
cme	1		
comex	1		
cse	1		
fox	1		
fse	1		
hkse	1		
ipe	1		
jse	1		
klce	1		
klse	1		
liffe	1		
lme	1		
lse	1		

Fig. 1

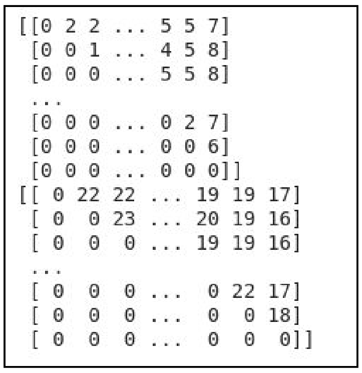
We convert the csv data to word form using pandas , and then using numpy library we get the array representation of all the words present and the labels present separately stored in the form of a vector representation . The below figure, Fig.2 shows the word array:



```
['amex' 'ase' 'asx' 'biffex' 'bse' 'cboe' 'cbt' 'cme' 'comex' 'cse' 'fox'
 'fse' 'hkse' 'ipe' 'jse' 'klce' 'klse' 'liffe' 'lme' 'lse' 'mase' 'mise'
 'mnse' 'mose' 'nasdaq' 'nyce' 'nycsce' 'nymex' 'nyse' 'ose' 'pse' 'set'
 'simex' 'sse' 'stse' 'tose' 'tse' 'wce' 'zse' 'adb-africa' 'adb-asia'
 'aibd' 'aid' 'anrpc' 'asean' 'atpc' 'bis' 'cipec' 'comecon' 'ec' 'eca'
 'ecafe' 'ece' 'ecla' 'ecsc' 'ecwa' 'efta' 'eib' 'emcf' 'escap' 'euratom'
 'fao' 'gatt' 'gcc' 'geplacea' 'iaea' 'iata' 'icco' 'ico-coffee'
 'ico-islam' 'ida' 'iea' 'iisi' 'ilo' 'ilzsg' 'imco' 'imf' 'inro' 'stoph'
 'strougal' 'subroto' 'suharto' 'sumita' 'suominen' 'takeshita' 'tamura'
 'tavares-moreia' 'thatcher' 'timar' 'tinsulanonda' 'tiwari' 'toernaes'
 'toman' 'tsovolas' 'vancsa' 'venkataraman' 'vera-la-rosa' 'verity'
 'villanyi' 'vlatkovic' 'volcker' 'von-weizsaecker' 'vranitzky' 'waldheim'
 'wali' 'walsh' 'wang-bingqian' 'wardhana' 'wasim-aun-jaffrey' 'wilson'
 'wise' 'yeutter' 'young' 'yu-kuo-hua' 'zak' 'zhao-ziyang' 'zheng-tuobin'
 'afghanistan' 'albania' 'algeria' 'american-samoa' 'andorra' 'angola'
 'anguilla' 'antigua' 'argentina' 'aruba' 'australia' 'austria' 'bahamas'
 'bahrain' 'bangladesh' 'barbados' 'belgium' 'belize' 'benin' 'bermuda'
 'bhutan' 'bolivia' 'botswana' 'brazil' 'british-virgin-islands' 'brunei'
 'bulgaria' 'burkina-faso' 'burma' 'burundi' 'cameroon' 'canada'
 'cape-verde' 'cayman-islands' 'central-african-republic' 'chad' 'chile'
 'china' 'colombia']
```

Fig. 2

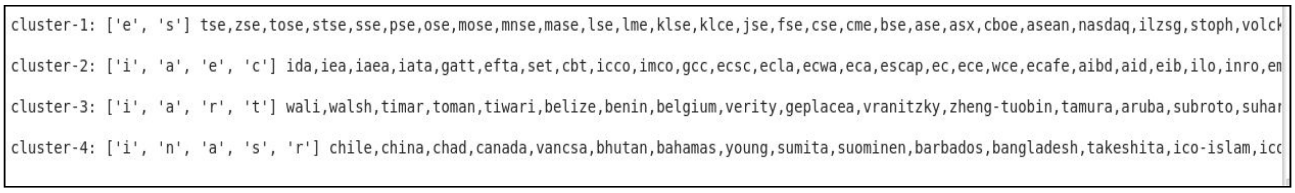
Once we have the word array, we calculate the similarity index values between the document pairs using distance package. Since we have 156 documents , we get a 156*156 similarity_index array which is represented in Fig.3 below: (We print only first few elements of this 156*156 matrix as everything cannot be printed in a page)



```
[[0 2 2 ... 5 5 7]
 [0 0 1 ... 4 5 8]
 [0 0 0 ... 5 5 8]
 ...
 [0 0 0 ... 0 2 7]
 [0 0 0 ... 0 0 6]
 [0 0 0 ... 0 0 0]]
[[ 0 22 22 ... 19 19 17]
 [ 0  0 23 ... 20 19 16]
 [ 0  0  0 ... 19 19 16]
 ...
 [ 0  0  0 ...  0 22 17]
 [ 0  0  0 ...  0  0 18]
 [ 0  0  0 ...  0  0  0]]
```

Fig. 3

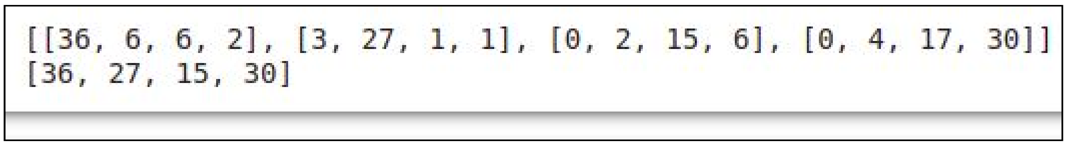
Once we have the similarity index matrix, we use the clustering algorithm to get the clusters initially, then get the frequent itemsets and we then normalize the clusters to get the required number of clusters along with their frequent itemsets. The clusters are shown in the below image, Fig 4.



```
cluster-1: ['e', 's'] tse,zse,tose,stse,sse,pse,ose,mose,mnse,mase,lse,lme,klse,klce,jse,fse,cse,cme,bse,ase,asx,cboe,asean,nasdaq,ilzsg,stoph,volck

cluster-2: ['i', 'a', 'e', 'c'] ida,iea,iaea,iata,gatt,efta,set,cbt,icco,imco,gcc,ecsc,ecla,ecwa,eca,escap,ec,ece,wce,ecafe,aibd,aid,eib,ilo,inro,em

cluster-3: ['i', 'a', 'r', 't'] wali,walsh,timar,toman,tiwari,belize,benin,belgium,verity,geplacea,vranitzky,zheng-tuobin,tamura,aruba,subroto,suhar

cluster-4: ['i', 'n', 'a', 's', 'r'] chile,china,chad,canada,vancsa,bhutan,bahamas,young,sumita,suominen,barbados,bangladesh,takeshita,ico-islam,icc
```

Fig. 4

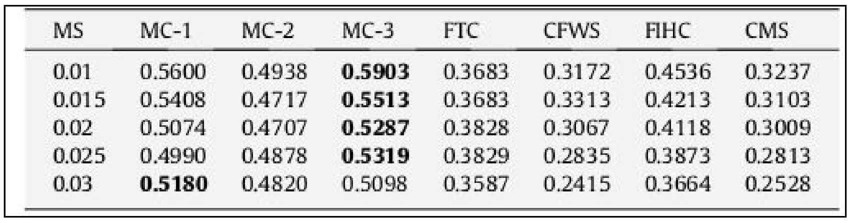
Now we have the normalized cluster, last step is the evaluation of our implementation done. We use the F-1 score as our metric for evaluation , so initially we get the matrix representation to calculate accuracy using F-1 metric. We then sum up the diagonal elements in the matrix, as these are the elements that are clustered correctly and divide with the total number of itemsets which gives the accuracy of our project. We get this accuracy as 69%. The matrix which describes actual value vs predicted value is shown in the form of a 4*4 matrix in Fig. 5(2nd line only corresponds to diagonal elements)



```
[[36, 6, 6, 2], [3, 27, 1, 1], [0, 2, 15, 6], [0, 4, 17, 30]]
[36, 27, 15, 30]
```

Fig. 5

We now study the improvement that we suggested for the existing Text clustering algorithm using Maximum Clustering algorithm. The improvement we suggested was for the case when the similarity index values are equal for deciding the maximum similarity index value, we have assigned priority for this case. The accuracy that is received for the Reuters-21578 dataset in the paper [1] using Maximum Clustering is 59%. This is shown in the figure below, Fig. 6 . But we have pre processed the dataset and only taken 4 categories instead of 6 that was suggested in the paper[1] , so the same algorithm stated in the paper works better for our dataset and gives 66.6% accuracy for the dataset we have taken.



MS	MC-1	MC-2	MC-3	FTC	CFWS	FIHC	CMS
0.01	0.5600	0.4938	**0.5903**	0.3683	0.3172	0.4536	0.3237
0.015	0.5408	0.4717	**0.5513**	0.3683	0.3313	0.4213	0.3103
0.02	0.5074	0.4707	**0.5287**	0.3828	0.3067	0.4118	0.3009
0.025	0.4990	0.4878	**0.5319**	0.3829	0.2835	0.3873	0.2813
0.03	**0.5180**	0.4820	0.5098	0.3587	0.2415	0.3664	0.2528

Fig. 6

After we add the improvement to the existing algorithm and we get an accuracy of 69% for the same dataset , suggesting that the changes that we have made are an improvement to the existing algorithm.

IV. Conclusions :

In this paper, we have conducted a study on the clustering of texts using frequent mining and some various other algorithms that we have defined to cluster the documents. We have tried to improvise on the existing Maximum Clustering algorithm for document clustering and have achieved to do so. For the implementation of the algorithm to divide the texts with similarity values into various clusters, we have introduced a concept of priority to improve the clustering accuracy. This priority concept is helpful when the similarity indexes are equal and we need to find the maximum similarity value between 2 documents. For evaluation, we have used F-1 score as our metric to get the accuracy and also showed that for the Reuters dataset, the accuracy increases with the improvements that we have done.

References :

1. Wen Zhang, Taketoshi Yoshida, Xijin Tang, Qing Wang, “Text clustering using frequent itemsets” , ELSEVIER-2010.
2. Benjamin C.M. Fung, Ke Wang, Martin Ester , “Hierarchical Document Clustering Using Frequent Itemsets”.
3. Bjornar Larsen and Chinatsu Aone, “Fast and Effective Text Mining Using Linear-time Document Clustering”
4. Michael Steinbach, George Karypis, Vipin Kumar, “A Comparison of Document Clustering Techniques”
5. JIAWEI HAN, JIAN PEI, YIWEN YIN, RUNYING MAO, “Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach”, 2004 Kluwer Academic Publishers.
6. Florian Beil, Martin Ester, and Xiaowei Xu, “Frequent Term-Based Text Clustering”
7. Yanjun Li, Soon M. Chung, John D. Holt , “Text document clustering based on frequent word meaning sequences”, ELSEVIER-2007
8. Yannis Haralambous and Philippe Lenca, “Text Classification Using Association Rules, Dependency Pruning and Hyperonymization” .
9. Oren Zamir and Oren Etzioni, “Web Document Clustering: A Feasibility Demonstration”
10. Rakesh Agrawal, Tomasz Imielinski, Arun Swami, “ Mining Association Rules between Sets of Items in Large Databases”