A FREQUENT DOCUMENT MINING ALGORITHM WITH CLUSTERING

Mr. Rakesh Kumar Soni
Department of Information Technology
Technocrats Institute of Technology, RGPV
Bhopal, M.P., India
er06rakeshsoni@gmail.com

Prof. Neetesh Gupta
Department of Information Technology
Technocrats Institute of Technology, RGPV
Bhopal, M.P., India
gupta_neetesh81@yahoo.com

Prof. Amit Sinhal
Department of Information Technology
Technocrats Institute of Technology, RGPV
Bhopal, M.P., India
amit_sinhal@rediffmail.com

Abstract

Now days, finding the association rule from large number of item-set become very popular issue in the field of data mining. To determine the association rule researchers implemented a lot of algorithms and techniques. FP-Growth is a very fast algorithm for finding frequent item-set. This paper, give us a new idea in this field. It replaces the role of frequent item-set to frequent sub graph discovery. It uses the processing of datasets and describes modified FP-algorithm for sub-graph discovery. The document clustering is required for this work. It can use self-similarity function between pair of document graph that similarity can use for clustering with the help of affinity propagation and efficiency of algorithm can be measure by F-measure function.

Keywords: clustering, document-graph, FP-growth, graph mining, frequent sub graphs clustering.

1 INTRODUCTION

Data mining techniques is used to extracting information from that data. Many researchers developed a lot of algorithm and techniques for finding useful information in the database. Frequent item-set mining is a core data mining operation and has been extensively studied over last decade. It plays an essential role in many important data mining tasks. Algorithms for frequent item-set mining form the basis for algorithms for a number of other mining problems, including association rule mining, correlations mining, and mining sequential and emerging patterns. Association rule mining technique is very effective data mining technique to finding the useful hidden information in the data, its aim to extract correlation, frequent pattern, association by transaction database or other data repository. This rules generated by datasets and it derives by measurement of support and confidence of every role, which define the frequency of that role. Association rule is a rule which is depends on the association relationship of objects and items. For example data item interrelation ship which is occurs simultaneous with other data items. This rule is calculated by data and association calculated with the help of probability. Share market and recommended system etc. are its practical applications.

To concept of document based clustering, association rule mining is very effective algorithm and useful approach. It utilizes that FP-Growth approach discovers frequent patterns and modified it to frequent sub graph discovery. Originally this algorithm is design to frequent item-set mine in market basket analysis. In this paper, analyse FP-Growth approach to document clustering. For graph mining, change in FP-Growth algorithm which discover frequent connected graph and perform better for thick connected graph.

This Paper consists of five main sections. The second section describes the Basic Theory of this work. The frame work of this approach is describes in third section, Experimental Result analysis is forth section and finally section five is devoted to conclusion and future work.

2 Basic Theory

The association rule mining [2] approaches were established to determine frequent item-sets in market basket datasets [4]. The two most common approaches to association rule mining are Frequent Pattern growth (FP-growth) [3] and apriori [5]. The apriori algorithm uses prior information of frequent item-sets to produce the candidates for larger frequent item-sets. It relies on interactions between item-sets and subsets. If an item-set is frequent, then all of its subsets must also be frequent. But generating candidates and checking their support at each level of iteration can become expensive. FP-growth introduces a different approach here. Instead of generating the candidates, it compresses the database into a compact tree form, known as the FP-tree, and extracts the frequent patterns by traversing the tree.

In this work, Exchanged the concept of the frequent item-set in to frequent sub graph. However, there are some famous sub graph discovery systems like FSG (Frequent Sub graph Discovery) [6], gSpan (graph-based Substructure pattern mining) [7], DSPM (Diagonally Sub graph Pattern Mining) [8], and SUBDUE [9]. It does allow us to believe that the concept of the creation of document-graphs and discovering frequent sub graphs to execute a sense-based clustering is currently feasible. All these systems deal with multiple aspects of efficient frequent sub graph mining. Most of them have been tested on real and artificial datasets of chemical compounds. Yet none of them has been applied to the mining of text data. It's developed a graph-based approach for document clustering using FP-growth. Since in this system the documents are represented by graphs, made the necessary changes to the FP-growth algorithm so that instead of generating frequent item-sets it can generate frequent sub graphs.

3 Frame Work of This Approach

Our overall method is briefly portrayed in this section. The system is divided into four major parts: Document Pre-processing, FP-Growth approach, Document clustering and Similarity Measures. Detailed mechanisms are described in the following subsections.

3.1 Document Pre-processing

The work use stemming to repossess the relationship between the keywords of a document. In stemming, words are related to each other based on different semantic relationships among them. This Paper use the relation of keywords, which is a super-ordinate or IS-A ("a kind of") relationship. This work believes that, for sense-based document clustering, this relation is more suitable than the others because it provides the generalization of a concept or a word. It builds a document-graph by traversing the hierarchy of the keywords up to Entity. Once there have all of the document-graphs, create a Master Document Graph (MDF) [11] based on the traversal of all of the keywords up to the topmost level.

Now use the MDF for a number of reasons. Instead of taking the entirety of Words as the background knowledge. Use MDF that contains the ontology related to the keywords only. This allows us to focus on the area of the Words which is applicable to our dataset. Another purpose of the use of MDF is that it can facilitate the use of the DFS codes. The code is produced by traversing the MDF in Depth First Search (DFS) order. This gives us the DFS traversal order (DFS code) for the entire MDF. The DFS codes of the edges of the MDF are applied to the edges of the document-graphs. Thus, the DFS codes of the edges of the MDF help us to mark each edge of the document-graphs in a DFS order specific manner. Therefore, the same subgraph acting in two document-graphs can be identified by the DFS codes of the edges of that subgraph. Additionally, the MDF helps in testing the connectivity between nodes during the subgraph discovery process.

The graph representation of the documents to table consisting of the list of edges appearing in each document.

Table 1: Modified FP-growth algorithm [6]

```
Input Document graphs' database DGB,
 Master Document graph MDF,
 Minimum support min_sup

Output     Frequent sub graphs

1. Create a list called transactionDGB for all DGF∈ DGB
2. Create headerTable for all edge ai ∈ MDF
3. FilterDGB(transactionDGB,headerTable, min_sup)
4. FPTreeConstrcutor()
5. FPMining() // seeTable 2 for details
```

This helps us to fit the frequent sub graph discovery task to the frequent item-set discovery problem of the market basket data analysis. Each edge of a document-graph is considered to be an item of a transaction, a sub graph is considered to be an item-set and each document is considered to be a transaction.

3.2 FP-growth Approach for Frequent Sub graph Discovery

The original FP-growth algorithm, when applied to our problem generates a set of frequent edges which do not necessarily constitute to a connected sub graph. Generation of all possible frequent patterns not only outputs all possible frequent sub graphs but also generates a lot of overhead in the form of all possible sets of frequent, but disconnected, edges. This causes unnecessary costs during the mining of the document-graphs as only looking for frequent sub graphs and these sets of frequent disconnected edges tolerate no useful information for the document clustering. The time and space required to generate and store these disconnected frequent edges have negative impact on the overall performance of the FP-growth approach. To overcome these problems it restricts the discovery of frequent item-sets in the FP-growth approach. Instead of discovering all possible frequent combination of edges we only discover the frequent edges that founds to a graph. We also control the maximum size of the frequent sub graph by the SP Threshold value.

The shape of modified FP-growth approach is given in Table 1. The method Filter DGB reconstructs both the lists transaction DGB and header Table by eliminating the infrequent edges based on the minimum support provided by the user. It sorts the edges of transaction DGB and header Table in descending order by support. After this step, the top of the header covers the most frequent edges and the bottom contains the least frequent ones. The edges at the top level of the header table are the representative edges of the topmost levels of our MDF. Since these edges denote very abstract relationships between synsets [10], they appear in too many document-graphs implying that they are not good candidates for clustering. In contrast, edges at the bottom of the header table are least frequent and denote relations between very specific concepts in the MDF. Since they appear in very few document-graphs, they too provide less information for clustering. This motivated us to crop the edges a second time from the top and bottom of the header table before constructing the FP-tree. transactionDB is updated accordingly to reflect this change in the header table. After this refinement, create the FP-tree by calling the FPTreeConstructor() method. Later, FP-Mining () generates the frequent sub graphs by traversing the FP-tree.

Table 2: Algorithm for FP-Mining: *FPMining().[6]*

```
Input    FP-tree FpT
  FList headerTable
  Frequent pattern a (initially, a = null)

Output Frequent sub graphs b
1.          if FpT contains a single path
2.                if (FpT.length> SPThreshold)
3.  Delete (FpT.length - SPThreshold)   number of edges from the top of FpT.
4.  for each combination (denoted as b) of the nodes in path T
5.  if (isConnected(b,a) = = true)// see Table 3 for details
6. generate b∪a, support = MIN (support of nodes inb)
7.          else for each a_i in the headerTable of FpT
8.   generate pattern b = a_i∪a with support =a_i.support;
9.   if (isConnected(b, a_i) = = true)
10. construct b's conditional pattern base and use it to build b's conditional FP-tree Treeb and b's conditional
header table headerTableb
11.          if (Tree b ≠ ∅ )
12.             FP-Mining(Treeb, headerTableb,b);
```

In the original FP-growth approach a node from an FP-tree indicates an item. In that case, a node of the FP-tree contains the DFS-code of an edge. If the original FP-growth approach is directly used in graph-mining, it does not guarantee the connectivity property of a subgraph. It can generate a disconnected set of edges in a representation of a frequent subgraph. The difference between the original FP-growth approach and our modified approach is that the system crops the single paths of the FP-tree and maintains connectivity of edges for discovering frequent sub graphs. The modified FP-growth algorithm for our sub graph discovery method is described in Table 2.

If at any point, a single-path is encountered in the FP-tree, we crop nodes from the top of the single path based on the user-provided threshold SPThreshold. Removing a node from the single path refers to eliminating the corresponding edge denoted by that node. SPThreshold provides control to the number of combinations of edges appearing in a single path. Depending on the connectivity (Table 3) of the edges, a combination of the edges may or may not generate a frequent sub graph. Let b be a sub graph for which a single path is generated by traversing the branches of the FP-tree ending with b. this say that the discovery of the newly joint frequent sub graphs is conditional on the frequent sub graph b (Table 2). The supports of these new sub graphs are determined by the support of b before the merging (step 6 of Table 2).

The depth of the MDF can reach up to 18 levels, which is the maximum height of the hierarchy of Words. Since our document-graphs contain hundreds of edges, the depth of the FP- tree can reach up to hundreds of levels depending on the number of edges in a document-graph. Conventional FP-growth generates all possible subsets of a single path for every edge-conditional FP-tree. Instead of accepting all possible combinations of a single path it only keeps the combinations of edges that generate connected frequent sub graphs. Whenever a single path is encountered in the FP-tree (or recursively generated conditional FP-trees), each of its combinations is generated and checked to make sure that it follows the connectivity constraint. This is done by first taking one of the edges from the combination of the single path and then adding it to a list called connectedList (Table 3). In the Iterate () method, a neighboring edge list, neighborListi is created for an edge i using the MDF. The neighborListi is checked to see if it contains any edge from the combination of the conditional FP-tree’s single path. If there are such edges it followed the same procedure for all of them until no new edges from the combination of the single path are added to the connectedList. At the end, the size of the connectedList is compared with the size of the combination (in step 4 of Table 3). If both of their sizes are the same, then the whole combination must be connected generating a subgraph a. Then an attempt is made by step 6 through 9 of Table 3 to combine the subgraph b with subgraph a. The method isConnected() returns true if a and b can be merged together to generate a connected subgraph. It should be noted that the isConnected method checks for the connectivity of one combination at a time. If the edges under consideration of this combination do not compose a connected subgraph, but compose multiple disconnected sub graphs, then some other combinations of the single path will generate these smaller connected sub graphs. So, do not lose disjoint but smaller connected sub graphs of a larger disconnected combination of the single path.

Table 3: Checking connectivity: *isConnected (b, a).[6]*

```
Input Combination of edges, b
  Frequent pattern, a
Output Returns true if b and a composes a connected subgraph, otherwise returns false.
Global Variable connectedList
Method    isConnected(b , a)
1.        connectedList = null;
2.        edge = the first edge of  b;
3.        Iterate (edge, b);  // is b connected
4.        if (connectedList.size ≠ b.size)
5.                return false;
6.        for each edge e_i in  b  // is a and b connected
7.                edge = the first edge in b
8.                if (isConnected(edge, a)  = = true)
9.                        return true;
10.     return false;
Method    Iterate (edge, subset)
11.     connectedList = connectedList ∪ edge
12.  neighbors = all incoming and outgoing edges of edg
13.  for each edge e_i in neighbours
14.  if  (subset  contains  e_i &&  connectedList does  not contain e_i )
15.          Iterate(e_i, subset)
```

Additionally, control the single path length by using SPThreshold so that our FP-growth approach performs faster. The length of the single path can be as large as the number of all the edges appearing in a document-graph. Taking each combination of a large single path and checking the connectivity constraints is computationally intensive. The construction of the FP-tree forces the edges with higher frequencies to appear at the top of the FP-tree. So, it is more likely that nodes at the top levels of the FP-tree indicate edges at more abstract levels of the MDF. After observing that a higher abstraction level of the MDF does not provide enough information for reliable clustering, we restricted the generation of combination of a single path of the FP-tree to a certain length.

(*SPThreshold*)

When he mines the FP-tree, he starts from the edges appearing the least in the pruned header table. Thus, if he reaches a single path of length greater than SPThreshold, he prunes the upper part of the single path above the threshold and generates each combination of the lower part only. This mechanism prunes nodes of single paths of the FP-tree at the upper levels which are representative of top level edges of the MDF.

3.3 Document Clustering

This used the discovered frequent sub graphs to cluster the document-graphs. These sub graphs can be viewed as concepts appearing frequently within the documents. If it evaluate the similarity of documents based on the co-occurrence of frequent sub graphs, and then use these similarity values to cluster the documents, it will get a sense-based clustering of the text documents. He uses Hierarchical Agglomerative Clustering (HAC) [12] for the clustering phase of that work. Now it implements the Group Average method to cluster the documents where the distance between two clusters is defined by the average distance between points in both clusters.

A number of similarity measures exist that can be used to find the closest or most distant pair of documents to merge during HAC. Among them, the cosine measure [13] is the most frequently used one. It penalizes less in cases where the number of frequent sub graphs on each document differs significantly. Since the cosine measure focuses more on the components in the documents and is not influenced by the document length, it has been used widely in document clustering.

3.4 Similarity Measures

In this work chose this measure to compute the similarity between two document-graphs (DG1 and DG2) based on the frequent sub graphs (FS) appearing in them:

$$\text{Similarity}_{\text{cosine}}\ (\text{DG1}, \text{DG2}) = \frac{\text{count}\ (\text{FS}(\text{DG1}) \cap (\text{DG2}))}{\sqrt{\text{count}(\text{FS}(\text{DG1})) * \text{count}(\text{FS}(\text{DG2}))}}$$

(1)

To cluster the documents use a dissimilarity matrix which stores the dissimilarity between every pair of document-graphs using the formula, dissimilarity = 1– similarity. The value of dissimilarity can range from 0 to 1.

It can use self-similarity in work in place of similarity function this self-similarity can be define as K-means algorithm adopts the widely used similarity measurement Cosine coefficient, namely equation [1].Because k-means needs to compute on the vector space model, for X = (x1,…, xn) and Y = (y1, …, yn), the component form of their similarity measurement can be expressed as:

$$\boldsymbol{S(X,Y)} = \frac{\sum_{i=1}^{n} x_i y_i}{\left(\sum_{i=1}^{n} x_i^2\right)^{\frac{1}{2}} \left(\sum_{i=1}^{n} y_i^2\right)^{\frac{1}{2}}}$$

(2)

Where n is the number of the features in the whole vector space. Similar with k-means, AP (CC) and SAP (CC) also use Cosine coefficient (equation (2)) as the similarity between two documents. However, unlike k-means, a document in AP (CC) and SAP (CC) does not need to be represented into the whole vector space, but only into its own vector space. Therefore the similarity measurement computation complexity of the latter two algorithms is reduced greatly in respect to the one of k-means. The self-similarities of AP (CC) are defined as:

$$\boldsymbol{s(l,l) = \min_{1\le i,j\le N, i\ne j}\{s(i,j)\} - \varphi(\max_{1\le i,j\le N, i\ne j}\{s(i,j)\} - \min_{1\le i,j\le N, i\ne j}\{s(i,j)\})}$$

(3)

Where φ is an adjustable factor?

This similarity can be used for clustering with help of affinity propagation and efficiency of algorithm can be measured be F-Measure function using precision and recall value and Entropy function.

4 Experimental Result Analysis

4.1 Experimental Results of Modified FP Growth Approach

Table 5 contains some of our experimental results with the subset of 768 documents from 7 different groups of items. The avg. min_sup was kept 49.2 and we selected 568 items from the middle of the 768 documents to use them in our FP-growth approach.

Table 5- Parameters of Mining FP-Tree and Generate association rules.

No. of Groups	Min_sup	FP-Tree Storage (Bytes)	FP-Tree updates	FP-Tree Nodes	Generating Time (Seconds)	T-Tree Storage (Bytes)	No. of Frequent Sub graph Nodes
10	2.0	580	59	24	0.39	8992	639
25	5.0	602	186	25	0.44	8516	607
50	10.0	1284	364	56	0.34	8800	623
101	20.2	1702	759	75	0.27	8284	590
200	40.0	2516	1534	112	0.33	8816	629
568	113.6	3770	4473	169	0.3	8360	596
768	153.6	4386	6090	197	0.3	8460	604

The Table 5 shows the outputs of FP-Growth algorithm in which taken a combination of groups like 10, 25, 50, 101, 200, 568, 768, and so on. This table shows the variation of minimum support, FP-Tree Storage in bytes, FP-tree updates after pruning nodes in the header table, No. of FP-Tree Nodes, time for generating association rules in seconds, T-Tree storage in Bytes and last No. of frequent Sub graph Nodes

Here take an experimental result of table 5 and plot a graph of minimum support value of all the no. of groups which show the efficiency of minimum support of FP-Growth algorithm.

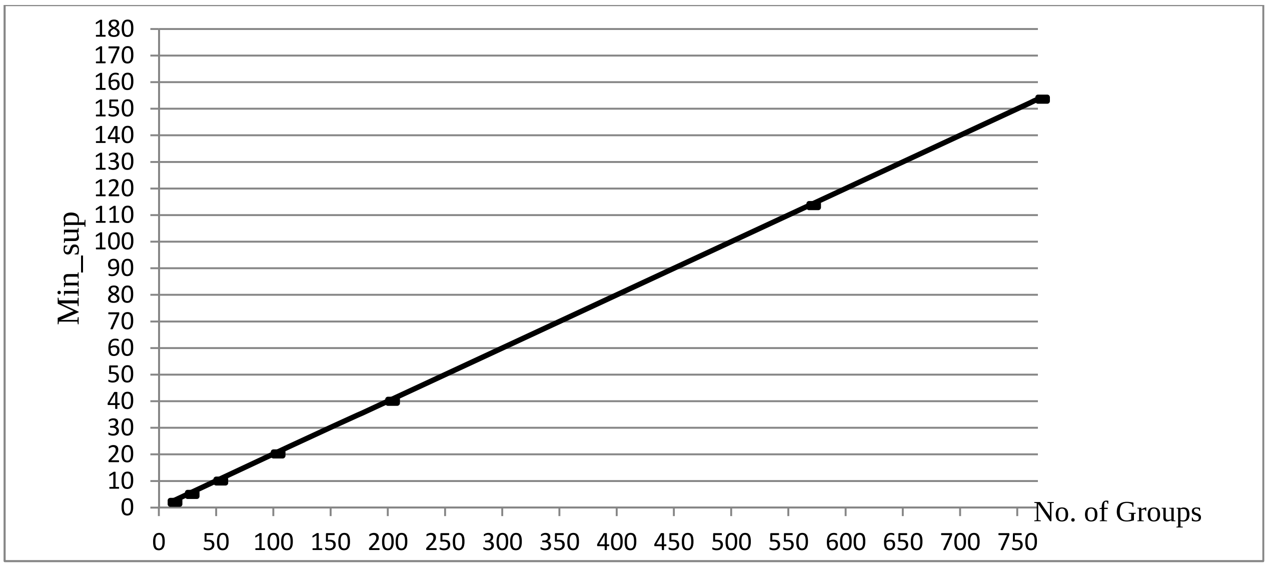

Figure 5.1- Graph for minimum support of FP-Growth Algorithm.

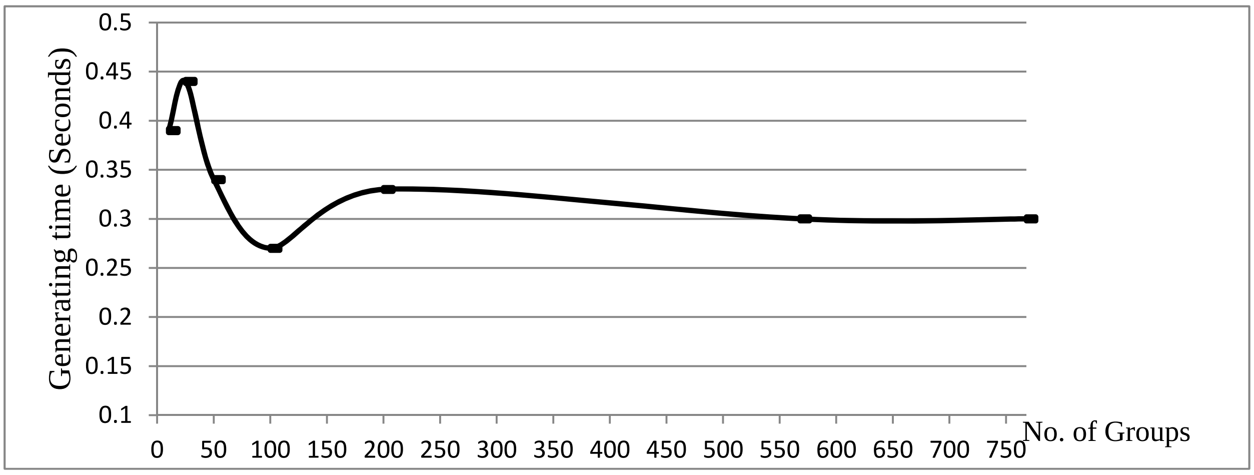

Figure 5.4- Graph for Generating time in second of FP-Growth.

4.2 Experimental Results of Document Clustering

We used the frequent sub graphs find by the FP-growth approach in the earlier section to cluster the corresponding documents took dissimilar numbers of keywords from each dataset for the clustering. To examine the accuracy of our clustering, we analyse the results with the commonly used frequency-based clustering technique.

Table 6 contains our experimental results of document clustering with affinity propagation for different documents. Here table shows 8 different data files clustering outputs of documents.

Table 6 - Parameters of document files generated in clustering.

No. of Files	5	10	20	30	40	50	60	70
No. of Identified	1	2	6	8	12	15	18	19
Data point (net similarity)	5.7 e^{-306}	4.46	12.69	20.32	25.99	31.88	39.07	47.55
Preference of selected exemplars	0	0	4.91	6.62	9.60	9.90	14.26	15.68
No. of iterations	126	131	114	108	121	114	122	120
Elapsed Time	2.5	2.72	2.22	2.14	2.39	2.29	2.64	2.41
No. of cluster	1	2	6	8	12	15	18	19
Fitness (net similarity)	5.7 e^{-306}	4.46	17.66	26.95	35.59	41.79	53.33	63.23

Table 6 shows the overall parameters of document clustering which shows that when clustering method apply on the document file, the number of iteration and elapsed time in this method will be constant and overall process take average iteration for all clustering is 119.5 and average time is 2.41375 second for every clustering.

Now consider result with respect to files and elapsed time, graph of elapsed time and documents is shown below.

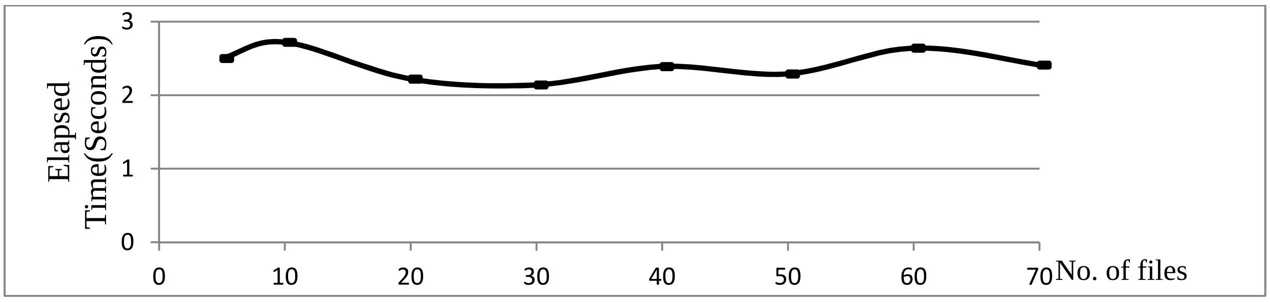

Figure 5.5- Elapsed time in second for documents in clustering.

Now consider result with respect to files and no of iteration, graph of no. of iteration and document files is shown below.

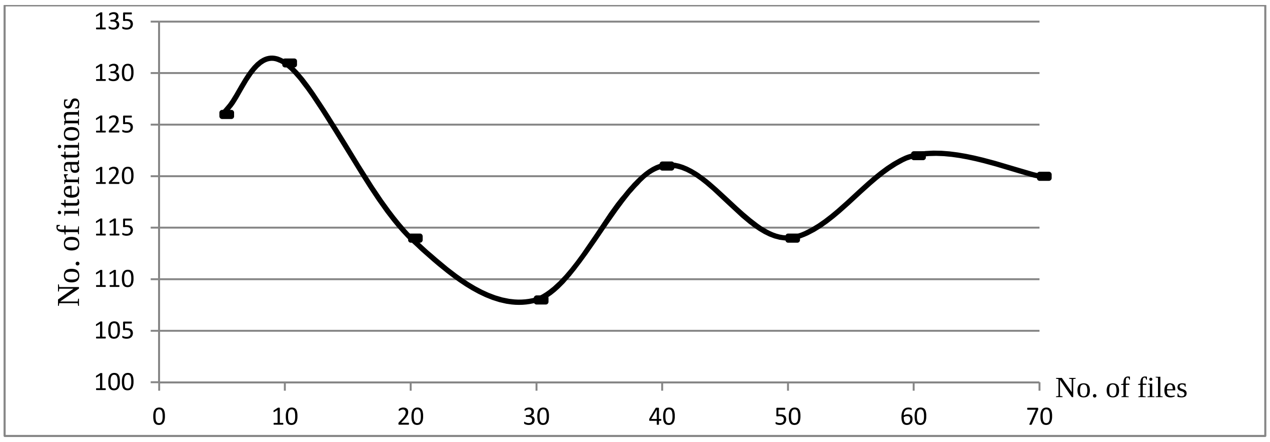

Figure 5.6 - No. of iterations for documents in clustering.

For the number of documents, if we raise the number of keywords, the MDF has the affinity to contain more edges. More keywords better illuminate the concept of the documents. In such case, the edges in the mid-level of the MDF will have more relations between them. As a result, more sub graphs will appear in the middle of the MDF. So, our methodology has the affinity to take the advantage of attachment of keywords by determining more sub graphs and using them in the clustering. It requires inclusion of thousands of keywords for better clustering.

5 CONCLUSION

In Data mining document clustering is very active research area.to find suitable information system much suitable idea has been implemented in document clustering. It is very challenging task to find human-like clustering. In this work, a graph based clustering with affinity propagation. That find a new way to clustering document based more on the keywords they contain document based clustering techniques mostly depend on the keywords. The work modifying the FP-mining algorithm to find the frequent sub graph with clustering affinity propagation in graph.

It can evaluate some techniques for computing the combination of large scale paths which can improve the performance of mining algorithms. In future will plan to study more application related to that feature and try to implement batter way and improve their performance.

6 REFERENCES

[1] Renchu Gaun, Xiaohu Shi,Maurizio Marchese, Chen yang and Yanchun Liang, “Text Clustering with seeds Affinity Propagation” European Commission under grant No. 155776-EM-1-2009-1-IT-ERAMUNDUS-ECW-L12, 2009.

[2] ABM Rezbaul Islam, Tae-Sun Chung, “an improved frequent pattern tree based association rule mining tree”,IEEE,2011

[3] Jiawei Han, Jian Pei, and Yiwen Yin, “Mining Frequent Patterns without Candidate Generation”, SIGMOD Paper ID: 196, 2000

[4] Han, J., and Kamber, M., “Data Mining: Concepts and Techniques”, 2nd edition, ISBN, 2006.

[5] Agrawal, R. and Srikant, R., “Fast Algorithms for Mining Association Rules”, Proc. of the 20th Int'l Conference on Very Large Databases, Santiago, Chile, pp.487-499, (September 1994).

[6] Monika Akbar, Rafal A. Angryk, “Frequent Pattern-Growth Approach for Document Organization”, Napa Valley, California, USA, 2008.
[7] Kuramochi, M., and Karypis, G., “An efficient algorithm for discovering frequent sub graphs”, IEEE Trans. on KDE, pp.1038-1051, 16, 9 (2004).
[8] Yan, X., and Han, J., “gSpan, Graph–Based Substructure Pattern Mining”, Proc. IEEE Int’l Conf. on Data Mining ICDM, Maebashi City, Japan, pp.721–723, (November 2002).
[9] Cohen, M., and Gudes, E., “Diagonally sub graphs pattern mining”, Workshop on Research Issues on Data Mining and Knowledge Discovery proceedings, pp.51–58, (2004).
[10] Ketkar, N., Holder, L., Cook, D., Shah, R., and Coble, J. “Subdue: Compression-based Frequent Pattern Discovery in Graph Data”, ACM KDD Workshop on Open-Source Data Mining, pp.71-76, (August 2005).
[11] http://www.wordnet.princeton.edu
[12] Hossain, M. S., and Angryk, R. A., “GDClust: A Graph-Based Document Clustering Technique”, IEEE ICDM Workshop on Mining Graphs and Complex Structures, USA, pp.417-422, (2007).
[13] Xiaojin Zhu, jerryzhu@cs.wisc.edu, “Clustering”, CS769 Spring Advanced Natural Language Processing, 2010.
[14] Anna Huang, “Similarity Measures for Text Document Clustering”, proceedings of the New Zealand Computer Science Research Student Conference, 2008
[15] Kaufman, L. and Rousseeuw, P. J., “Finding Groups in Data: an Introduction to Cluster Analysis”, John Wiley & Sons, (1990).