Frequent Pattern-Growth Approach for Document Organization

Monika Akbar
Department of Computer Science
Virginia Tech, Blacksburg, VA 24060, USA.
amonika@cs.vt.edu

Rafal A. Angryk
Department of Computer Science
Montana State University, Bozeman, MT 59717, USA.
angryk@cs.montana.edu

ABSTRACT
In this paper, we propose a document clustering mechanism that depends on the appearance of frequent senses in the documents rather than on the co-occurrence of frequent keywords. Instead of representing each document as a collection of keywords, we use a document-graph which reflects a conceptual hierarchy of keywords related to that document. We incorporate a graph mining approach with one of the well-known association rule mining procedures, FP-growth, to discover the frequent subgraphs among the document-graphs. The similarity of the documents is measured in terms of the number of frequent subgraphs appearing in the corresponding document-graphs. We believe that our novel approach allows us to cluster the documents based more on their senses rather than the actual keywords.

Categories & Subject Descriptors:
H.4.0 Information Systems, INFORMATION SYSTEMS APPLICATIONS, General

General Terms: Algorithms

Keywords: Clustering, Document-Graph, FP-growth, Graph mining, WordNet, Frequent Subgraph Clustering.

1. INTRODUCTION
Organizations and institutions around the world store data in digital form. As the number of documents grows, a robust way of extracting information from them becomes more important. It is vital to have a reliable way to cluster massive amounts of text data. In this paper, we present a new way of clustering documents using frequent subgraphs that represent the sense of a document.

Most of the current document mining techniques, commonly known as bag-of-words (BoW) approaches [1], depend on the frequency of the keywords to cluster the documents. In these models, a document is represented as a vector whose elements are the keywords with their frequencies. In most cases, this is not sufficient to represent the concept of the document and can result in an ambiguous result. There is no way to keep the information concerning the relations among the keywords in each document. Many words have multiple meanings, and once they are stored as individual units, it is hard to identify the specific meaning of a keyword in that document. In the BoW representation of the documents, the overall concept of a document does not become apparent. Therefore, it may provide low quality clustering.

Using an algorithm for association rule mining [2] to cluster documents based on their concepts is a fairly new approach. We utilized the Frequent Pattern growth (FP-growth) approach [3] that discovers frequent patterns, and modified this approach so that it can discover frequent subgraphs. Originally, this algorithm was designed to mine frequent itemsets in the domain of market basket analysis [4]. Unfortunately, it was not designed with graph mining in mind and does not efficiently mine frequent subgraphs. In this paper, we analyze the FP-growth approach for graph mining with the aim of efficient document clustering. To make the existing FP-growth algorithm suitable for graph mining, we changed the algorithm so that it discovers frequent connected subgraphs and performs better even for bushy document-graphs.

The rest of the paper is organized as follows. Section 2 describes the background of this work. The overall system is portrayed in section 3. Some illustrative experimental results are discussed in section 4. We conclude the paper in section 5.

ONISW'08, October 30, 2008, Napa Valley, California, USA.

2. BACKGROUND
The association rule mining [2] strategies were developed to discover frequent itemsets in market basket datasets [4]. The two most popular approaches to association rule mining are Frequent Pattern growth (FP-growth) [3] and Apriori [5]. The Apriori algorithm uses prior knowledge of frequent itemsets to generate the candidates for larger frequent itemsets. It relies on relationships between itemsets and subsets. If an itemset 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 costly. 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 our work, we replaced the concept of the frequent itemset to frequent subgraph. However, there are some well-known subgraph discovery systems like FSG (Frequent Subgraph Discovery) [6], gSpan (graph-based Substructure pattern mining) [7], DSPM (Diagonally Subgraph Pattern Mining) [8], and SUBDUE [9]. These works allow us to believe that the concept of the construction of document-graphs and discovering frequent subgraphs to perform a sense-based clustering is currently feasible. All these systems deal with multiple aspects of efficient frequent subgraph 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. We developed a graph-based approach for document clustering

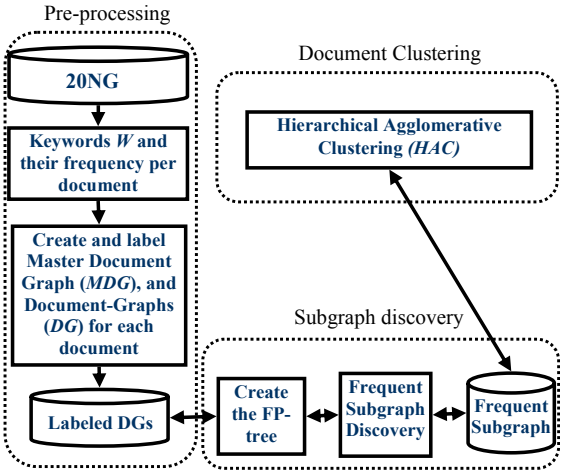

Figure 1: Overall mechanism of graph-based document clustering using FP-growth

using FP-growth. Since in our system the documents are represented by graphs, we made the necessary changes to the FP-growth algorithm so that instead of generating frequent itemsets it can generate frequent subgraphs.

3. SYSTEM OVERVIEW

Our overall system is briefly portrayed in Figure 1. The system is divided into three major parts: Pre-processing, Subgraph discovery and Document clustering. The detailed mechanisms are described in the following subsections.

3.1 Pre-processing

We use WordNet [10] ontology to retrieve the relationship between the keywords of a document. In WordNet, words are related to each other based on different semantic relationships among them. We have used the hypernymy relation of noun keywords, which is a super-ordinate or IS-A ("a kind of") relationship. We believe 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. All nouns in WordNet 2.1 are merged to a single topmost synset (i.e.,{entity}). We construct a document-graph by traversing the hypernym hierarchy of the keywords up to *Entity*. Once we have all of the document-graphs, we create a Master Document Graph (MDG) [11] based on the traversal of all of the keywords up to the topmost level. We used the MDG for a number of reasons. Instead of taking the entirety of WordNet as the background knowledge we use MDG that contains the ontology related to the keywords only. This allows us to concentrate on the area of the WordNet which is relevant to our dataset. Another reason why we use the MDG is that it can facilitate the use of the DFS codes. The code is generated by traversing the MDG in Depth First Search (DFS) order. This gives us the DFS traversal order (DFS code) for the entire MDG. The DFS codes of the edges of the MDG are applied to the edges of the document-graphs. Thus, the DFS codes of the edges of the MDG help us to mark each edge of the document-graphs in a DFS order specific manner. Therefore, the same subgraph appearing in two document-graphs can be identified by the DFS codes of the edges of that subgraph. Additionally, the MDG helps in checking the connectivity between nodes during the subgraph discovery process.

We transformed our graph representation of the documents to a table consisting of the list of edges appearing in each document.

Table 1: Modified FP-growth algorithm

Input	Document graphs' database *DB* Master Document graph *MDG* Minimum support *min_sup*
Output	Frequent subgraphs $subGraph_j$
1.	Create a list called *transactionDB* for all $DG_i \in DB$
2.	Create *headerTable* for all edge $a_i \in MDG$
3.	*FilterDB(transactionDB, headerTable, min_sup)*
4.	*FPTreeConstrcutor()*
5.	***FPMining()*** // seeTable 2 for details

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

3.2 FP-growth Approach for Frequent Subgraph 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 subgraph. Generation of all possible frequent patterns not only outputs all possible frequent subgraphs 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 we are only looking for frequent subgraphs and these sets of frequent disconnected edges bear 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 we restricted the discovery of frequent itemsets in the FP-growth approach. Instead of discovering all possible frequent combination of edges we only discover the frequent edges that constitutes to a graph. We also control the maximum size of the frequent subgraph by the

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

Input	FP-tree *T* FList *headerTable* Frequent pattern α (initially, $\alpha = null$)
Output	Frequent subgraphs β
1.	if *T* contains a single path
2.	if (*T.length*> *SPThreshold*)
3.	delete (*T.length* - *SPThreshold*) number of edges from the top of *T*.
4.	for each combination (denoted as β) of the nodes in path *T*
5.	if (***isConnected***(β, α) = = true) // see Table 3 for details
6.	generate $\beta \cup \alpha$, *support* = MIN *(support of nodes in* β*)*
7.	else for each a_i in the *headerTable* of *T*
8.	generate pattern $\beta = a_i \cup \alpha$ with *support* = a_i.*support*;
9.	if (***isConnected***(β, a_i) = = true)
10.	construct β's conditional pattern base and use it to build β's conditional FP-tree $Tree_\beta$ and β's conditional header table $headerTable_\beta$
11.	if ($Tree_\beta \neq \varnothing$)
12.	***FP-Mining***$(Tree_\beta, headerTable_\beta, \beta)$;

SPThreshold value.

The outline of our modified FP-growth approach is given in Table 1. The method *FilterDB* reconstructs both the lists *transactionDB* and *headerTable* by removing the infrequent edges based on the minimum support provided by the user. It sorts the edges of *transactionDB* and *headerTable* in descending order by support. After this step, the top of the header contains 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 MDG. Since these edges represent 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 represent relations between very specific concepts in the MDG. Since they appear in very few document-graphs, they too provide less information for clustering. This motivated us to prune 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, we created the FP-tree by calling the *FPTreeConstructor()* method. Later, *FPMining()* generates the frequent subgraphs by traversing the FP-tree.

In the original FP-growth approach a node from an FP-tree indicates an item. In our 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 our system prunes the single paths of the FP-tree and maintains connectivity of edges for discovering frequent subgraphs. The modified FP-growth algorithm for our subgraph discovery process is described in Table 2. If at any point, a single-path is encountered in the FP-tree, we prune nodes from the top of the single path based on the user-provided threshold *SPThreshold*. Removing a node from the single path refers to removing the corresponding edge represented 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 subgraph. Let β be a subgraph for which a single path is generated by traversing the branches of the FP-tree ending with β. We say that the discovery of the newly combined frequent subgraphs is conditional on the frequent subgraph β (Table 2). The supports of these new subgraphs are determined by the support of β before the merging (step 6 of Table 2).

Table 3: Checking connectivity: *isConnected (β, α).*

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

The depth of the MDG can reach up to 18 levels, which is the maximum height of the hypernym hierarchy of WordNet. 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 we only keep the combinations of edges that generate connected frequent subgraphs. 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, $neighborList_i$ is created for an edge *i* using the MDG. The $neighborList_i$ 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 we 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 α. Then an attempt is made by step 6 through 9 of Table 3 to combine the subgraph β with subgraph α. The method *isConnected()* returns true if α and β 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 subgraphs, then some other combinations of the single path will generate these smaller connected subgraphs. So, we do not lose disjoint but smaller connected subgraphs of a larger disconnected combination of the single path.

Additionally, we 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 MDG. After observing that a higher abstraction level of the MDG 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 we mine the FP-tree, we start from the edges appearing the least in the pruned header table. Thus, if we reach a single path of length greater than *SPThreshold*, we prune the upper part of the single path above the threshold and generate 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 MDG.

3.3 Clustering

We used the discovered frequent subgraphs to cluster the document-graphs. These subgraphs can be viewed as concepts appearing frequently within the documents. If we evaluate the similarity of documents based on the co-occurrence of frequent subgraphs, and then use these similarity values to cluster the documents, we will get a sense-based clustering of the text documents. We use Hierarchical Agglomerative Clustering (HAC) [12] for the clustering phase of our work. We implemented 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 subgraphs 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. We have chosen this measure to compute the similarity between two document-graphs (*DG1* and *DG2*) based on the frequent subgraphs (*FS*) appearing in them:

$$Similarity_{cosine}(DG1, DG2) = \frac{count\left(FS(DG1) \cap FS(DG2)\right)}{\sqrt{count\left(FS(DG1)\right) \times count\left(FS(DG2)\right)}}$$

To cluster the documents we 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.

3.3.1 Evaluation Mechanism

To evaluate the clustering, we use the average Silhouette Coefficient (SC) [14]. The average Silhouette Coefficient generates values in the range [-1, 1], where a negative value is undesirable because it corresponds to the case in which the average dissimilarity of documents in the cluster is greater than the minimum average dissimilarity of documents in other clusters, and 1.0 indicates results with ideal separation between clusters. We compared our system with the traditional keyword-frequency-based (bag-of-tokens) clustering mechanism of vector space that uses the cosine similarity measure for the construction of the dissimilarity matrix.

4. EXPERIMENTAL RESULTS

We used two different subsets of the 20Newsgroups dataset [15] for our experiments: the first subset contains 500 documents from 5 groups and the second one consists of 2500 documents from 15 groups. In the next two subsections we analyze our results for subgraph discovery and clustering respectively.

4.1 FP-growth for Subgraph Discovery

Table 4 describes the impact of single path threshold (*SPThreshold)* in the total number of all *k*-edge subgraphs. The number of discovered subgraphs increases or remains the same with an increasing *SPThreshold*. The number remains the same when added new edges are not connected with the edges of the previous single path. Inclusion of these new edges in the single path does not result in additional connected frequent subgraphs. More and more large subgraphs are discovered as a result of increased *SPThreshold*. Table 5 gives a partial description of the highlighted row with *SPThreshold*=18 of Table 4. It shows the number of generated subgraphs from each row of the header table. Column **N(6)** of Table 5 shows that our subgraph discovery algorithm detected a total of 583 *6*-edge subgraphs. In contrast, Table 4 shows that **N(6)** is 480 when the single path threshold is 18. The numbers are different because **N(6)** of Table 4 indicates the number of unique subgraphs but **N(6)** of Table 5 indicates the total number of subgraphs discovered by different rows of the header table. In Table 5, some subgraphs discovered by the edge of the *i*-th row of the header table may already be encountered by a previous edge of row *j* of the header table. Therefore, the last row of Table 5 shows the total number of *k*-edge subgraphs detected by all the edges of the header table, where Table 4 shows how many of them are unique.

Table 4: Decomposition of subgraphs.

# of documents: 500, #of groups = 5, *min_sup = 5%*
N(k) = # of *k*-edge subgraphs
SPT = *SPThreshold*

SPT	..	N(6)	..	N(9)	..	N(17)	..	N(20)
10	..	473	..	425	..	0	..	0
12	..	479	..	450	..	0	..	0
14	..	480	..	450	..	0	..	0
16	..	480	..	456	..	122	..	0
18	**..**	**480**	**..**	**456**	**..**	**122**	**..**	**0**
20	..	480	..	456	..	114	..	28

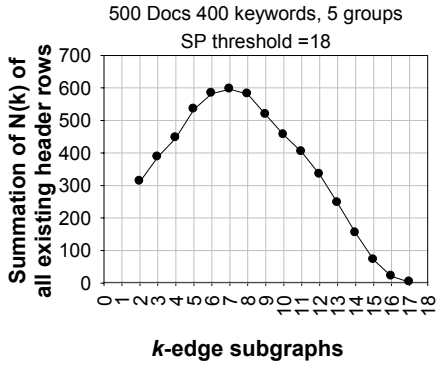

Figure 2: Discovered subgraphs of modified FP-growth.

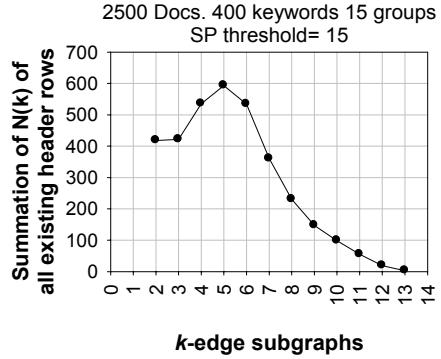

Figure 3: Discovered subgraphs of modified FP-growth.

Table 5: Subgraphs generated for the edges in the header table.

# of documents: 500, #of groups = 5, *min_sup = 5%*, *SPThreshold* = 18
i = index of the header table (0 indicates top of the header table)

i	N(2)	N(3)	N(4)	N(5)	N(6)	…	N(17)
…	…	…	…	…	…	…	…
308	5	5	7	16	23	…	0
…	…	…	…	…	…	…	…
302	14	9	0	0	0	…	0
…	…	…	…	…	…	…	…
16	19	26	30	31	30	…	0
…	…	…	…	…	…	…	…
12	8	14	17	19	19	…	0
…	…	…	…	…	…	…	…
Total	**312**	**387**	**446**	**534**	**583**	**…**	**3**

The actual behavior of the number of *k*-edge subgraphs discovered from Table 5 is shown in Figure 2. The number of *1*-edge frequent subgraphs is the same as the number of edges appearing in the header table after pruning. We start the plot from *2*-edge subgraphs. This number increases in large subgraphs, but after a certain point it starts to decrease since larger subgraphs become rare in the document-graphs. Figure 3 shows a similar plot for our second dataset with 2500 documents from 15 groups with 400 keywords. The single path threshold is 15 for Figure 3.

Figure 4 and Figure 5 show the corresponding runtime behaviors of our frequent subgraph discovery process using the FP-growth approach. Since the algorithm starts from the bottom of the existing header table with the least frequent edges, the X-axis of each of the plots indicates the sequence of edges from the bottom of the header table. The plots show the time required for discovering all subgraphs for each edge in the header table. The times plotted in the figures are

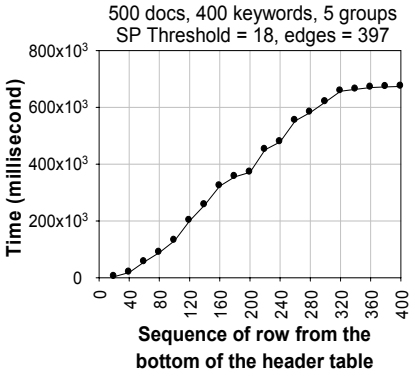

Figure 4: Running time for FP-Mining

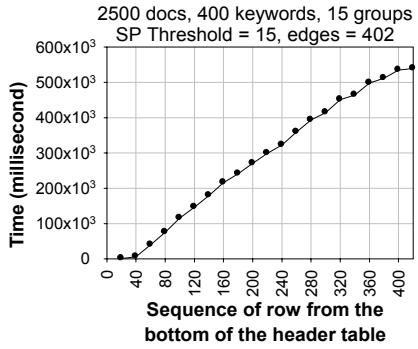

Figure 5: Running time for FP-Mining

Table 6: Average SC for 5 clusters.

# of documents: 500, #of groups = 5

Number of keywords	SC coefficient for 5 clusters		% Better $\left(\frac{X-Y}{X}\right)\times 100$
	FP-growth approach (*X*)	Traditional frequency based (*Y*)	
200	0.202190	0.054568	73.01
300	0.210349	0.052667	74.96
400	0.228002	0.024996	89.04
500	0.342921	0.021473	93.74
600	0.256827	0.035883	86.03

Table 7: Average SC for 15 clusters.

# of documents: 2500, #of groups = 15

Number of keywords	SC coefficient for 15 clusters		% Better $\left(\frac{X-Y}{X}\right)\times 100$
	FP-growth approach (*X*)	Traditional frequency based (*Y*)	
200	0.442671	0.009031	97.96
300	0.733246	-0.007208	100.98
400	0.430419	-0.014922	103.47
500	0.669504	-0.013861	102.07
600	0.562792	-0.028308	105.03

cumulative. The figures show that the execution time is almost linear in terms of time required to discover subgraphs for each edge of the header table. Moreover, at the end of execution the runtime becomes almost flat. This happens because edges at the top of the header table have higher chance of pointing to nodes at the top of the FP-tree. Nodes closer to the top of the FP-tree cause construction of smaller conditional FP-trees. As a result, the time required to discover the frequent subgraphs with edges from the top of the header table is relatively small.

4.2 Clustering

We used the frequent subgraphs discovered by the FP-growth approach of the previous section to cluster the corresponding document corpora consisting of 500 and 2500 documents respectively. We took different numbers of keywords from each dataset for the clustering. To investigate the accuracy of our clustering, we compared our clustering results with the commonly used frequency-based clustering technique.

Table 6 shows the comparison between the average Silhouette Coefficients using our approach and the traditional approach for the dataset with 500 documents. We calculated the average Silhouette coefficient for 5 clusters with different numbers of keywords from the same dataset. In every case with 500 documents, our approach provides better accuracy than the traditional keyword-frequency-based approach.

We show similar experimental results with 2500 documents from 15 groups in Table 7. Since the dataset is larger than the previous one, but provided keywords are not sufficient for clustering 2500 documents, the clustering accuracy (average SC) is very low for traditional vector based model. Moreover, sometimes the average Silhouette Coefficient is negative indicating inaccurate clustering using the traditional approach. In contrast, our approach shows good clustering accuracy even with small numbers of keywords. Table 7 shows that our clustering approach is 105.03% better than the

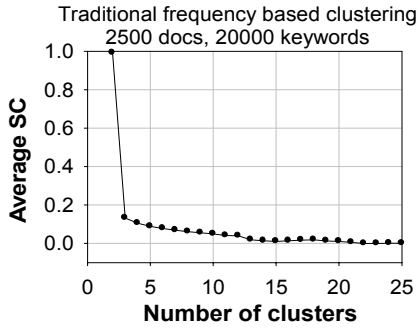

Figure 6: Average SC for traditional keyword frequency based clustering.

traditional approach when there are 15 clusters in 2500 documents with 600 keywords.

For the same number of documents, if we increase the number of keywords, the Master Document Graph (MDG) [11] has the tendency to contain more edges. More keywords better clarify the concept of the documents. In such cases, the edges in the mid levels of the MDG will have more connections between them. As a result, more subgraphs will appear in the middle of the MDG. So, our approach has the tendency to take the advantage of inclusion of keywords by discovering more subgraphs and using them for clustering. The traditional keyword-frequency-based approach does not exhibit this tendency if small amounts of keywords are added. It requires inclusion of thousands of keywords for better clustering. The average SC for a different number of clusters using the traditional approach for 2500 documents from 15 groups with 20,000 keywords is shown in Figure 6. With 20,000 keywords, the average SC was improved to only 0.0105 for 15 clusters. The highest average SC for the same dataset using our clustering mechanism was 0.733246 for 300 keywords (Table 7). Therefore, our clustering mechanism is able to cluster the documents more accurately, even from a low dimensional dataset.

5. CONCLUSION

Document clustering is a very active research area in data mining. Numerous ideas have been explored to find suitable systems that cluster documents more meaningfully. Even so, producing more intuitive and human-like clustering remains a challenging task. In our work, we have developed a sense-based clustering mechanism by employing a graph mining technique. We have introduced a new way to cluster documents based more on the senses they represent than on the keywords they contain. Traditional document clustering techniques mostly depend on keywords, whereas our technique relies on the keywords' concepts. We have modified the FP-growth algorithm to discover the frequent subgraphs in document-graphs. Our experimental results show that modified FP-growth can be used to find frequent subgraphs successfully, and that the clustering based on discovered subgraphs performs better than the traditional keyword-frequency-based clustering.

This work can be extended in a number of directions. Mining the FP-tree for larger single path lengths is still computationally expensive. We can generate some optimization techniques for computing the combinations of larger single paths which can improve the performance of the FP-growth approach. Additionally, devising an intelligent system to extract useful edges from the header table remains as an area for future research.

6. REFERENCES

[1] Lewis, D., "Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval", 10th European Conference on Machine Learning, Chemnitz, DE: Springer Verlag, Heidelberg, DE, (1998), 4-15.

[2] Agrawal, R., and Swami, T. I., "Mining Association Rules between sets of items in large databases", Proc. Of ACM SIGMOD, (1993), 207-216.

[3] Han, J., Pei, J., and Yin, Y., "Mining frequent patterns without candidate generation" Proc. 2000 ACM-SIGMOD Int. Conf. Management of Data, Dallas, TX, (2000), 1–12.

[4] Han, J., and Kamber, M., "Data Mining: Concepts and Techniques", 2nd edition, ISBN 9781558609013.

[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, (September 1994), 487-499.

[6] Kuramochi, M., and Karypis, G., "An efficient algorithm for discovering frequent subgraphs", IEEE Trans. on KDE, 16, 9 (2004), 1038-1051.

[7] Yan, X., and Han, J., "gSpan, Graph–Based Substructure Pattern Mining", Proc. IEEE Int'l Conf. on Data Mining ICDM, Maebashi City, Japan, (November 2002), 721–723.

[8] Cohen, M., and Gudes, E., "Diagonally subgraphs pattern mining", Workshop on Research Issues on Data Mining and Knowledge Discovery proceedings, (2004), 51–58.

[9] 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, (August 2005), 71-76.

[10] http://www.wordnet.princeton.edu

[11] Hossain , M. S., and Angryk, R. A., "GDClust: A Graph-Based Document Clustering Technique", IEEE ICDM Workshop on Mining Graphs and Complex Structures, USA, (2007), 417-422.

[12] Tan , P., Steinbach, M. and Kumar, V., " Introduction to Data Mining", 1st edition, ISBN: 9780321321367, Addison-Wesley, (2005).

[13] Salton, G., "Automatic Text Processing", Addison-Wesley Publishing Company, 1988.

[14] Kaufman, L. and Rousseeuw, P. J., "Finding Groups in Data: an Introduction to Cluster Analysis", John Wiley & Sons, (1990).

[15] http://people.csail.mit.edu/jrennie/20Newsgroups