Semantically-Guided Clustering of Text Documents via Frequent Subgraphs Discovery

Rafal A. Angryk[1], M. Shahriar Hossain[2], Brandon Norick[1]

[1] Department of Computer Science, Montana State University,
Bozeman, MT 59717, USA
[2] Department of Computer Science, Virginia Tech
Blacksburg, VA 24061, USA

angryk@cs.montana.edu, msh@cs.vt.edu, brandon.norick@msu.montana.edu

Abstract. In this paper we introduce and analyze two improvements to GDClust [1], a system for document clustering based on the co-occurrence of frequent subgraphs. GDClust (Graph-Based Document Clustering) works with frequent senses derived from the constraints provided by the natural language rather than working with the co-occurrences of frequent keywords commonly used in the vector space model of document clustering. Text documents are transformed to hierarchical document-graphs, and an efficient graph-mining technique is used to find frequent subgraphs. Discovered frequent subgraphs are then utilized to generate accurate sense-based document clusters. In this paper, we introduce two novel mechanisms called the Subgraph-Extension Generator (SEG) and the Maximum Subgraph-Extension Generator (MaxSEG) which directly utilize constraints from the natural language to reduce the number of candidates and the overhead imposed by our first implementation of GDClust.

Keywords: graph-based data mining, text clustering, clustering with semantic constraints.

1 Introduction

There has been significant increase in research on text clustering, natural language processing and textual information extraction in the last decade. Most of these techniques rely on searching for identical words and counting their occurrences. The major motivation for our approach comes from typical human behavior when people are given the task of organizing multiple documents. As an example, consider the behavior of a scientific book editor who is faced with the complicated problem of organizing multiple research papers into a single volume with a hierarchical table of contents. Typically, even papers from the same research area are written (1) in multiple writing styles (e.g., using “clusters” instead of “concentration points”), (2) on different levels of detail (e.g., survey papers versus works discussing a single algorithm) and (3) in reference to different aspects of an analyzed area (e.g., clustering of numeric versus categorical data). Instead of searching for identical words and counting their occurrences, as many well-known text clustering techniques do [3]-[5], the human brain usually remembers only a few crucial keywords and their abstract senses, which provide the editor with a compressed representation of the analyzed document. These keywords, discovered thanks to the expert's knowledge (replaced, in our case, by WordNet [6]-[7]), are then used by the book editor to fit a given research paper into a book's organization scheme, reflected by the table of contents.

In this paper we improve the GDClust [1], a system that performs frequent subgraph discovery from a text repository for document clustering. In our approach document similarity is evaluated by generating a graph representation of each document, where edges in the graph represent hypernym relationships of keywords and abstract terms. Thus, a document-graph represents the structure within the ontology, which is independent of its specific keywords and their frequencies. In [1], we have shown that GDClust, which relies less on the appearance of specific keywords, is more robust than the traditional vector space model-based clustering and represents an advanced method for organizing the numerous documents available to us on a daily basis. Here, we propose a novel Subgraph-Extension Generation (SEG) mechanism that significantly outperforms a well-known FSG [7] approach, used in the original implementation of GDClust [1]. Additionally, we enhanced our Subgraph-Extension Generation by introducing our Maximum Subgraph-Extension Generation (MaxSEG) mechanism which provides faster dissimilarity matrix construction with the cost of slightly deteriorated performance compared to our SEG approach.

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

2 Background

Graph models have been used in complex datasets and recognized as useful by various researchers in chemical domain [9], computer vision technology [10], image and object retrieval [11] and social network analysis [12]. Nowadays, there are well-known subgraph discovery systems like FSG (Frequent Subgraph Discovery) [2], gSpan (graph-based Substructure pattern mining) [13], DSPM (Diagonally Subgraph Pattern Mining) [14], and SUBDUE [15]. Most of them have been tested on real and artificial datasets of chemical compounds. None of them however, has been applied to the mining of text data.

Agrawal et al. [8] proposed the Apriori approach for frequent patterns discovery. It is an iterative algorithm, where candidates for larger frequent patterns are gradually grown from frequent patterns of a smaller size. We start from discovering frequent patterns of size k=1, and in the next iteration merge them into the candidates for k=2-size frequent patterns. After the candidates are created, we check for frequencies of their occurrences, and move on to the next iteration.

There had been extensive research work in the area of generating association rules from frequent itemsets [16–17]. There are also some transaction reduction approaches proposed by Han et al. [18]. We apply our own variation of mining multilevel association rules [18] for the frequent sense discovery process and utilize the Gaussian minimum support strategy to prune edges from multiple levels of the terms.

Our system (GDClust [1]) utilizes the power of using graphs to model a complex sense of text data. It constructs the document-graphs from text documents and a semantic lexicon, WordNet [6]-[7], and then applies an Apriori paradigm [8] to discover frequent subgraphs from them. GDClust uses frequent subgraphs as the representation of common abstract senses among the document-graphs. The benefit of GDClust is that it is able to group documents in the same cluster even if they do not contain common keywords, but still possess the same sense. The vector space model of document clustering cannot perform this sort of discovery [3]-[5].

The work we managed to find that is closest to our approach is a graph query refinement method proposed by Tomita et al. [19]. Their prototype depends on user interaction for the hierarchic organization of a text query. In contrast, we depend on a predefined ontology [6-7], for the automated retrieval of frequent subgraphs from text documents. GDClust offers an unsupervised system that utilizes an efficient subgraph discovery technique to harness the capability of sense-based document clustering.

3 System Overview

Our GDClust pipeline can be divided into three major components: (1) the Conversion Unit and (2) the Subgraph Discovery Unit and (3) the Clustering Unit.

The Conversion Unit is responsible for the conversion of each document to its corresponding graph representation. It utilizes the Word Vector Tool (WVTool) [20] for the retrieval of meaningful keywords from the documents. It uses the WordNet hypernymy hierarchy to construct the document-graphs. We utilized WordNet's noun taxonomy, which provides hypernymy-hyponymy relationships between concepts and allows the construction of a document-graph with up to 18 levels of abstractions. Our document-graph construction algorithm traverses up to the topmost level of abstractions of the keywords of a document to construct the corresponding document-graph. To incorporate natural language constraints and speed up the process of frequent subgraph discovery, we also construct a master document-graph (MDG) which is a merged document-graph containing connections between all the keywords of all the documents and their abstract terms. Section 3.2 describes how the MDG helps in faster generation of candidate subgraphs.

The Subgraph Discovery Unit discovers frequent subgraphs representing frequent senses from the generated document-graphs. The Clustering Unit constructs the dissimilarity matrix and clusters the documents utilizing the frequent subgraphs that were discovered by the Subgraph Discovery Unit. Sections 3.2 and 3.3 describe the subgraph discovery processes and the clustering mechanism used by our GDClust.

3.1 Candidate Subgraph Pruning Using Gassian Minimum Support

We use an Apriori paradigm, designed originally for finding frequent itemsets in market basket datasets [4], to mine the frequent subgraphs from the document-graphs. We utilize our Gaussian minimum support strategy to logically prune *1*-edge subgraphs from candidate list before generating any higher order subgraphs. At each iteration of the frequent subgraph discovery process, a *k*-edge candidate subgraph is generated by using the *(k-1)*-edge subgraphs of the candidate subgraph list L_{k-1}.

Since using WordNet results in a very large graph of all English nouns, we introduced the master document-graph (MDG) and propose a dynamic minimum support strategy in GDClust. We use the dynamic minimum support strategy to limit the number of candidate subgraphs with extremely abstract and very specific synsets. Since WordNet's ontology merges to a single term, the topmost level of abstraction is a common vertex for all of the generated document-graphs, yielding subgraphs that include the vertex with the topmost level of abstraction to be less informative for clustering. Moreover, terms near to the bottom of the hierarchy are less useful due to their rare appearance in the document-graphs causing them to be of little use for clustering purposes. Terms appearing within the intermediate levels of the taxonomy seem to be more representative clusters' labels than subgraphs containing terms at higher and lower levels.

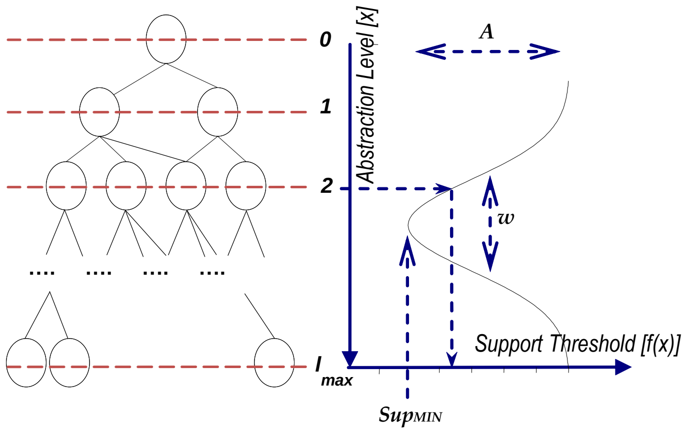

Fig. 1. Abstraction constrained Gaussian Minimum Support

Our dynamic minimum support strategy, based on Gaussain function used to model minimum support distribution is illustrated in Fig. 1. The hierarchy drawn in the figure indicates our master document-graph where the gray dots indicate the keywords. w is the width of the curve at $A/2$ and l_{max} is the maximum number of abstraction levels in the master document-graph. Since there are 18 levels of abstraction in WordNet's noun taxonomy, in our master document-graph $0 \le l_{max} < 18$. Our Gaussian model accepts only the keywords which have *frequency≥min_sup* (our predefined minimum support threshold) only when they appear at the mid level of our abstraction-based taxonomy. In the remaining cases, the model applies a gradual increase of minimum support at higher and lower levels. This model makes the mid-levels of the taxonomy formed by master document-graph more important. It also assumes, based on our observation, that the generated document-graphs contain a lot of frequent, but uninteresting subgraphs at the topmost level. At the same time, the document-graphs have distinct subgraphs at the bottom levels which are not frequent enough to carry significant meaning for the clustering purposes. The first group of subgraphs would generate large clusters with low inter-cluster similarity, and the second would generate a huge number of very small clusters. We apply the Gaussian dynamic minimum support strategy to prune *1*-edge subgraphs before the starting of generation of higher order subgraphs.

3.2 Semantically-guided Candidate Subgraph Generation

Our document-graph construction algorithm ensures that a document-graph does not contain more than one edge between two nodes. Additionally, the overall subgraph discovery concept ensures that the same subgraph does not appear more than once in a particular document-graph. All the edges and nodes of a document-graph are uniquely labeled. We developed a fast method to generate candidate subgraphs named Subgraph-Extension Generator (SEG). We have compared our approach with the original FSG-based [2] mechanism. Additionally, we enhanced our SEG to Maximum Subgraph-Extension Generator (MaxSEG) which generates fewer amounts of subgraphs and thus offers faster dissimilarity matrix construction during the clustering phase. The descriptions of FSG, SEG and MaxSEG approaches are as follows.

1) *FSG [7]*: In the FSG approach, a *(k+1)*-edge candidate subgraph is generated by combining two *k*-edge subgraphs where these two *k*-edge subgraphs have a common core subgraph [7] of *(k-1)*-edges.
This approach requires time-consuming comparisons between core subgraphs during the generation of a higher level subgraph. To avoid edge-by-edge comparisons, we assigned a unique code for each subgraph from the list of their edges' DFS-codes. This code is stored as the hash-code of the subgraph object. Therefore, checking two

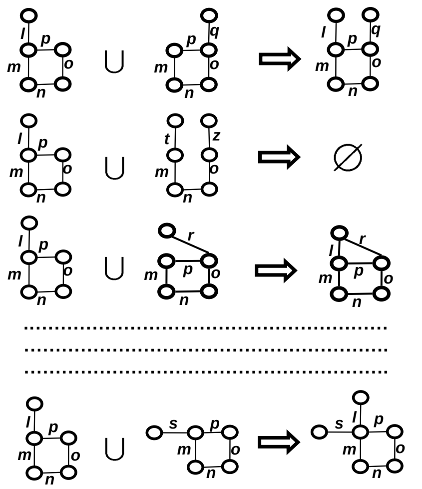

Fig. 2. Attempts to combine *lmnop* with other *5*-edge subgraphs of (L_5)

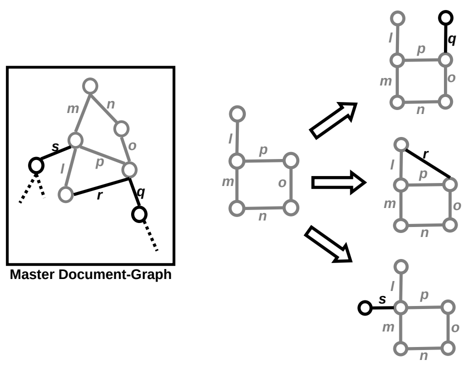

Fig. 3. *6*-edge Subgraph-Extension Generation for the *5*-edge subgraph *lmnop*

core subgraphs for equality has been reduced to a simple integer hash-code comparison. Although this approach is very fast for small graphs, it becomes inefficient for big document-graphs due to large number of blind attempts to combine *k*-edge subgraphs to generate *(k+1)*-edge subgraphs.

Consider an iteration in which we have a total of 21 *5*-edge subgraphs in the candidate list L_5. We try to generate *6*-edge subgraphs from this list. Consider the situation of generating candidates using one *5*-edge subgraph (e.g., *lmnop*) of L_5. The original FSG approach tries to combine all remaining 20 other subgraphs with *lmnop* but succeeds, let us assume, only in three cases. Fig. 2 illustrates that *lmnop* is successfully combined with only *mnopq*, *mnopr* and *mnops*. All 17 other attempts to generate a *6*-edge subgraph with *lmnop* fail because the *4*-edge core-subgraphs, analyzed in this case, do not match. Fig. 2 shows the attempts to generate good candidates for just one subgraph (*lmnop*). For all the subgraphs in L_5, there would be a total of $21{\times}20{=}420$ blind attempts to generate *6*-edge subgraphs. Some of these attempts would succeed, but most would fail to generate acceptable *6*-edge candidates. Although GDClust utilizes hash-codes of subgraphs and core-subgraphs for faster comparisons, it cannot avoid comparing a large number of hash-codes for all candidates using the FSG approach. We have reduced this number of comparisons by a significant degree by implementing our own and new Subgraph-Extension Generation approach.

2) *Subgraph-Extension Generator (SEG)*: Rather than trying a brute-force strategy of comparing all possible combinations (e.g., FSG), we use the master document-graph as the source of background knowledge to entirely eliminate the unsuccessful attempts while generating *(k+1)*-edge candidate subgraphs from *k*-edge subgraphs. We maintain a neighboring-edges' list for each *k*-edge subgraph and generate candidates for frequent higher order subgraphs by taking edges only from this list.

Fig. 3 shows the Subgraph-Extension Generation mechanism for subgraph *lmnop*, which can be compared with the FSG approach of Fig. 2. The gray edges of Fig. 3 are the edges of the *5*-edge subgraph which we want to extend to generate *6*-edge candidates. The black lines indicate the neighboring edges which extend the *5*-edge gray subgraph maintained in our MDG. The same instance is used for both Fig. 2 and Fig. 3 for an easy comparison. The neighboring-edges' list of *lmnop* contains edges *{q, r, s}*. Unlike in Fig. 2, in the example presented in Fig. 3, the Subgraph-Extension Generation technique does not try to blindly generate higher order subgraphs 20 times. Rather, it proceeds only three times, using the constraints coming from knowledge about the neighboring edges of *lmnop* in the MDG. It results in only three attempts to generate higher-order candidate subgraphs. None of these attempts fails to generate a candidate subgraph because the mechanism depends on the physical evidence of possible extension. Therefore, the Subgraph-Extension Generation of GDClust offers a novel knowledge-based mechanism that eliminates unnecessary attempts to combine subgraphs. All the generated candidate subgraphs that pass the minimum support threshold are entered into a subgraph-document matrix (analogous to term-document matrix of the vector-space model of document clustering). The subgraph-document matrix is used in the document clustering process later.

3) *Maximum Subgraph-Extension Generator (MaxSEG)*: In the MaxSEG approach, we keep only the largest frequent subgraphs and remove all smaller

subgraphs if they are contained in the higher order subgraphs. Any *k*-edge subgraph with support *s* is removed from the subgraph-document matrix if every *(k+1)*-edge frequent subgraph generated from it has the same *support=s*. If the *k*-edge subgraph generates at least one *(k+1)*-edge frequent subgraph that has *min_sup≤support<s* then we keep both the *k*-edge subgraph and the generated *(k+1)*-edge subgraphs.

In our implementation the SEG and the MaxSEG both require the same number of attempts to generate *(k+1)*-edge subgraphs from a *k*-edge subgraph, but they result in different numbers of subgraphs in the subgraph-document matrix. Consider that the *5*-edge subgraph *lmnop* of the example given in Fig. 3 appears in 20 document-graphs (*support=20*) and the minimum support threshold is 10. If every generated *6*-edge subgraph of the example has *support=20*, then we remove *lmnop* from the subgraph-document matrix and keep only the newly generated *6*-edge subgraphs. Now consider another situation where one of the generated *6*-edge subgraphs *lmnopq* has *support=15* and the other *6*-edge subgraphs have *support=20*. In this case, all of the *5*-edge and *6*-edge subgraphs *lmnop*, *lmnopq*, *lmnopr* and *lmnops* will remain in the subgraph-document matrix according to our MaxSEG approach. Therefore, the decision whether a *k*-edge subgraph will remain in the subgraph-document matrix or not is taken after generating all the *(k+1)*-edge subgraphs from this particular *k*-edge subgraph. This is why both SEG and MaxSEG approaches require the same number of attempts to generate higher order subgraphs, but the numbers of their resultant subgraphs are different.

3.3 Clustering and Evaluation

GDClust uses Hierarchical Agglomerative Clustering (HAC) [21] to gradually group the documents. We have chosen HAC because it facilitates the evaluation of our results from a broad range of generated clusters. The clustering unit constructs a dissimilarity matrix that contains dissimilarity values between every pair of document-graphs derived based on the number of common frequent subgraphs occurring in their respective document-graphs. We have used the cosine similarity measure [22] in GDClust because of its popularity in text clustering due to better stability of similarity values even when the comparing documents have significantly different sizes.

To evaluate the clustering of GDClust, we used both unsupervised and supervised evaluations of cluster validity. As an unsupervised evaluation we used the Average Silhouette Coefficient (ASC) [23]. We used entropy, purity and F-measure as supervised measures of cluster evaluation [24]. We compared our sense-based system with the traditional vector space model-based clustering mechanism (i.e., bag-of-words) that applies logarithmic normalization of keywords' frequency (TF-IDF) and uses the cosine similarity measure for the construction of the dissimilarity matrix. Additionally, we compared our sense-based system with a vector space model utilizing the background knowledge of WordNet (i.e., bag-of-concepts). With this clustering mechanism, we expand each term into its corresponding synsets (i.e., concepts) from WordNet, then we split the term's frequency between these synsets to obtain synset frequency values. Using these frequencies we compute values which are analogous to TF-IDFs in the term based model, and then these are used with the cosine similarity measure for the construction of the dissimilarity matrix. The same hierarchical agglomerative clustering algorithm as GDClust is used in the vector space model based systems to keep our experiments comparable.

4 Experimental Results

In our experiments, we used all 19997 documents of the 20 Newsgroups [25] dataset. The supervised evaluation of clustering is dependent on the class labels provided with the dataset. Some of the 20 class labels of the 20 Newsgroups dataset can be combined together to form a higher level group. As a result, the class labels provided with the dataset may not match the clusters well and therefore our supervised evaluation must carry some imperfection. Table 1 shows the list of the 20 Newsgroups, partitioned to 6 classes more or less according to subject matter as recommended in [25]. We used these 6 classes of 19997 documents for our supervised evaluation of clustering.

Table 1. 20 Newsgroups dataset [25].

Class	# of Docs	Original Newsgroups' Labels
1	5000	comp.graphics, comp.os.ms-windows.misc, comp.sys.ibm.pc.hardware, comp.sys.mac.hardware, comp.windows.x
2	4000	rec.autos, rec.motorcycles, rec.sport.baseball, rec.sport.hockey
3	4000	sci.crypt, sci.electronics, sci.med, sci.space
4	3000	talk.politics.misc, talk.politics.guns, talk.politics.mideast
5	2997	talk.religion.misc, alt.atheism, soc.religion.christian
6	1000	misc.forsale

4.1 Performance and Accuracy Analysis of the Subgraph Discovery Process

This section provides the experimental results of GDClust using the original FSG, Subgraph-Extension Generation (SEG), and Maximum Subgraph-Extension Generation (MaxSEG) approaches. Since the FSG approach is very slow compared to

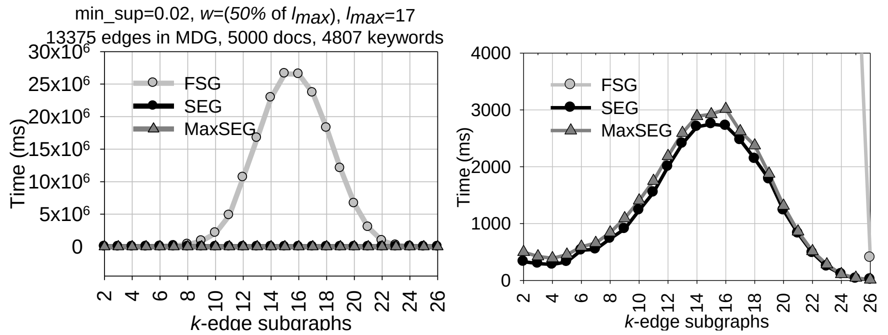

Fig. 4 Comparison between FSG, SEG, and MaxSEG. We changed the scale in Fig. 4 (b) to show the actual behaviors of SEG and MaxSEG ((b) is a close-up of (a)).

our SEG and MaxSEG mechanisms, we used only a subset of 5000 documents to show the comparison between approaches in Fig. 4. We constructed the stratified subset of 5000 documents by randomly selecting 25% of the documents from each of the 6 groups of Table 1. We show results with the complete 20 Newsgroups dataset in Section 4.2.

The improved GDClust offers the new SEG and MaxSEG approaches for the frequent subgraph discovery process which outperforms the existing FSG strategy. Due to the speed of SEG and MaxSEG, the lines drawn for them appear linear and flat in comparison to the light-gray line of the FSG approach (Fig.4(a)), although the actual behaviors of SEG and MaxSEG are not linear (Fig.4(b)). The curves maintain their hat-like shape, typical of the Apriori approaches, but due to the scale necessary to show the FSG results it is not clearly visible in Fig. 4 (a).

Table 2. Number of attempts to generate k+1-size candidates from the k-size freq. subgraphs, where *k* is the number of iteration in Apriori.

k	**FSG**	**SEG and MaxSEG**	**% of Attempts Saved**
1	---	---	---
2	11990	184	98.47%
3	7656	289	96.23%
4	6320	406	93.58%
5	8742	654	92.52%
6	18360	1150	93.74%
7	41412	2006	95.16%
8	92112	3386	96.32%
9	195806	5532	97.17%
10	416670	8954	97.85%
11	827190	13850	98.33%
12	1441200	19855	98.62%
13	2124306	25973	98.78%
14	2661792	31117	98.83%
15	*2861172*	*34365*	*98.80%*
16	2697806	35341	98.69%
17	2263520	34106	98.49%
18	1649940	30565	98.15%
19	1031240	25268	97.55%
20	539490	19035	96.47%
21	231842	12952	94.41%
22	78680	7810	90.07%
23	20306	4098	79.82%
24	3782	1827	51.69%
25	462	665	34.63%
26	30	185	23.33%

Table 3. Numbers of Frequent Subgraphs, actually discovered in each iteration (*k*) of all 3 implementations: FSG, SEG, and MaxSEG

k	**FSG and SEG**	**Max SEG**	**% of Subgraphs Reduced by Max SEG**
1	110	86	21.82%
2	88	72	18.18%
3	80	68	15.00%
4	94	81	13.83%
5	136	112	17.65%
6	204	164	19.61%
7	304	236	22.37%
8	443	338	23.70%
9	646	469	27.40%
10	910	605	33.52%
11	1201	733	38.97%
12	1458	832	42.94%
13	1632	877	46.26%
14	*1692*	*846*	*50.00%*
15	1643	782	52.40%
16	1505	696	53.75%
17	1285	590	54.09%
18	1016	468	53.94%
19	735	350	52.38%
20	482	241	50.00%
21	281	150	46.62%
22	143	83	41.96%
23	62	40	35.48%
24	22	16	27.27%
25	6	5	16.67%
26	1	1	0.00%

The SEG and MaxSEG approaches of GDClust outperform the FSG approach by a high magnitude due to the lower number of attempts used to generate higher order subgraphs by avoiding blind attempts. Table 2 shows that in every case SEG or MaxSEG saved a huge percentage of blind attempts generated by the FSG approach. The SEG and MaxSEG approaches saved 98.8% of the attempts while generating *15*-edge subgraphs from frequent *14*-edge subgraphs. Since *14*-edge subgraphs are the most frequent ones (Table 3), obviously the number of attempts to construct 15-edge subgraphs from *14*-edge subgraphs reaches the maximum for the FSG approach. Also, Fig. 4 shows that all the curves reach their peaks near *15*-edge subgraphs.

The numbers of the detected frequent *(k+1)*-edge subgraphs are the same for FSG and SEG because both methods generate complete sets of frequent subgraphs. However, they use different mechanisms to construct higher order subgraphs with different numbers of attempts. MaxSEG also generates same number of *(k+1)*-edge subgraphs from the list of *k*-edge subgraphs during its execution. But after extending a *k*-edge subgraph, MaxSEG removes it from the subgraph-document matrix if this particular *k*-edge subgraph has generated all *(k+1)*-edge subgraphs having the same support as this *k*-edge subgraph. This is the reason why numbers of generated left subgraphs are different in MaxSEG approach than the FSG or SEG. Fig. 4 (b) shows that the SEG approach is slightly faster than the MaxSEG approach. This is due to the fact that MaxSEG requires additional checks to verify the supports of the newly generated *(k+1)*-edge subgraphs in order to remove their immediate *k*-edge parent subgraph. Table 3 shows the exact number of the detected frequent *k*-edge subgraphs by the SEG and the number of maximum subgraphs detected by the MaxSEG. It also shows the percentage of the subgraphs removed by the MaxSEG approach. It shows that the MaxSEG approach removes 50% of the 14-edge subgraphs during the generation of *15*-edge subgraphs

Although the MaxSEG approach removes a lot of smaller subgraphs, our results show that it does not cause the same decrease of clustering quality. Table 4 shows that FSG, SEG and MaxSEG result in much better clustering than the vector space models. Note that while SEG offers faster execution than FSG, the clustering results are the same since they both generate the same set of subgraphs. Additionally, although the MaxSEG approach removes approximately 45% of all the total discovered subgraphs, it is not penalized during the clustering process in this ratio. Table 4 shows that the F-measure for MaxSEG is just slightly lower than the F-measure of the SEG approach. Since MaxSEG reduces the number of subgraphs, it offers faster construction of the dissimilarity matrix during the clustering phase compared to the FSG and SEG driven approaches. Table 4 also shows the side-by-side comparison between dissimilarity matrix construction times using different approaches.

Table 4. Dissimilarity Matrix construction times and the results of clustering presented via Supervised Measures: entropy, purity and F1.

	Time (sec)	**Supervised Evaluations**		
		Entropy	**Purity**	**F-measure**
FSG	768	0.84	0.81	0.77
SEG	768	0.84	0.81	0.77
MaxSEG	462	0.87	0.81	0.76

Traditional VSM	160	2.46	0.26	0.28
Concept VSM	95	2.46	0.25	0.26

Our GDClust-based approach to document clustering shows more accurate results than the vector space models of document clustering. Table 5 shows experimental results with different numbers of total keywords. It shows that the purity and F-measure have the tendency to become better and better while using our SEG and MaxSEG-based document clustering with an increasing number of keywords. In contrast, the vector space model based approaches do not guarantee more accurate results with inclusion of more keywords. This suggests that the GDClust is more robust than the vector space model based approaches.

Table 5. Results of clusterings, based on the different numbers of keywords.

Min.	**Num. of**	**SEG**		**MaxSEG**		**Traditional VSM**		**Concept VSM**	
TFIDF	**Keywords**	**Purity**	**F-meas.**	**Purity**	**F-meas.**	**Purity**	**F-meas.**	**Purity**	**F-meas.**
0.50	964	0.68	0.70	0.69	0.72	0.27	0.30	0.28	0.34
0.45	1340	0.70	0.71	0.70	0.72	0.28	0.30	0.29	0.33
0.40	1867	0.71	0.73	0.70	0.72	0.27	0.31	0.29	0.30
0.35	2544	0.73	0.73	0.73	0.73	0.27	0.31	0.27	0.29
0.30	3443	0.77	0.76	0.75	0.73	0.26	0.30	0.26	0.27
0.25	4807	0.81	0.77	0.81	0.76	0.26	0.28	0.25	0.26

4.2 Clustering Results with the Complete 20 Newsgroups Datasets [25]

In this subsection, we describe the clustering results with the complete 20 Newsgroups dataset of 19997 documents. Table 6 shows the entropy, purity, F-measure and Average Silhouette Coefficients (ASC) by the hierarchical agglomerative clustering of the 20 Newsgroups dataset. It shows that the best clustering is found using the SEG approach. Although the MaxSEG approach offers slightly faster dissimilarity matrix construction than the SEG based clustering, SEG dominates over MaxSEG in terms of accuracy. The traditional vector space model based approach does not show good structures. ASC=0.04 for 6 clusters indicates that the vector space model fails to provide clear separation between clusters. Furthermore, the concept based vector space model provides even worse separation between clusters, which indicates that the inclusion of background knowledge alone is not enough to provide good results

Table 6. Measures of clustering validity.

	Entropy	**Purity**	**F-measure**	**ASC**
SEG	0.84	0.81	0.78	0.76
MaxSEG	0.85	0.81	0.77	0.63
Traditional VSM	2.47	0.25	0.26	0.04
Concept VSM	2.47	0.25	0.21	-0.07

5 Conclusions

GDClust presents a valuable technique for clustering text documents based on the co-occurrence of frequent senses in documents. The approach offers an interesting, sense-based alternative to the commonly used vector space model for clustering text documents. Unlike traditional systems, GDClust harnesses its clustering capability from the frequent senses discovered in the documents. It uses graph-based mining technology to discover frequent senses. Unlike chemical compounds, our document-graphs may contain thousands of edges which results in a slow generation of frequent subgraphs during the discovery process using pre-existing graph mining techniques. We have introduced the Subgraph-Extension Generation and the Maximum Subgraph-Extension Generation techniques of frequent subgraph generation, which outperform the previous FSG strategy by a high magnitude by taking advantage of the constraints coming from our knowledge about natural-language. We have shown that our proposed approaches perform more accurately than a vector space model based system. GDClust is an automated system and requires minimal user interaction for its operations. In the future, we want to develop a more intelligent system to automatically determine the dynamic minimum support curve from the dataset and incorporate subgraph-weights in terms of size and importance to provide more accurate clustering results.

References

1. M.S. Hossain, R. Angryk, "GDClust: A Graph-Based Clustering Technique for Text Documents", Proceedings of the 7th IEEE International Conference on Data Mining (ICDM-IEEE '07), Workshop on Mining Graphs and Complex Structures, Omaha, NE, USA, October 2007, pp. 417-422.
2. M. Kuramochi and G. Karypis, "An efficient algorithm for discovering frequent subgraphs," IEEE Trans. on KDE, vol. 16, no. 9, pp. 1038-1051, 2004.
3. F. Sebastiani, "Machine learning in automated text categorization," ACM Comp. Surveys, vol. 34, no. 1, pp. 1-47, Mar. 2002.
4. C. D. Manning and H. Schutze, Foundations of Natural Language Processing. Cambridge, MA: MIT Press, 1999.
5. C. Cleverdon, "Optimizing convenient online access to bibliographic databases," Information Survey and Use, vol. 4, no. 1, pp. 37-47, Apr. 1984.
6. Cognitive Science Laboratory Princeton University, "WordNet: A Lexical Database for the English Language," [Online]. Available: http://wordnet.princeton.edu/
7. G. Miller, R. Beckwith, C. FellBaum, D. Gross, K. Miller, and R. Tengi, Five papers on WordNet. Princeton, NJ: Princeton University, 1993.
8. R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules", in Proc. of the 20th Intl. Conf. on Very Large Data Bases, 1994, pp. 487–499.
9. R. N. Chittimoori, L. B. Holder, and D. J. Cook, "Applying the SUBDUE substructure discovery system to the chemical toxicity domain," in Proc. of the 12th Intl. FLAIRS Conf., 1999, pp. 90-94.
10. D. A. L. Piriyakumar and P. Levi, "An efficient A* based algorithm for optimal graph matching applied to computer vision," in GRWSIA-98, 1998.

11. D. Dupplaw and P. H. Lewis, “Content-based image retrieval with scale-spaced object trees,” SPIE: Storage and Retrieval for Media Databases, vol. 3972, pp. 253-261, 2000.
12. M. E. J. Newman, “The structure and function of complex networks,” SIAM Review, vol. 45, no. 2, pp. 167-256, 2003.
13. X. Yan and J. Han, “gSpan: graph-based substructure pattern mining,” in IEEE ICDM, 2002, pp. 721-724.
14. C. Moti and G. Ehud, “Diagonally Subgraphs Pattern Mining,” in 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery, 2004, pp. 51-58.
15. N. Ketkar, L. Holder, D. Cook, R. Shah, and J. Coble, “Subdue: Compression-based Frequent Pattern Discovery in Graph Data,” in ACM KDD Workshop on Open-Source Data Mining, 2005, pp. 71-76.
16. R. Agrawal, M. Mehta, J. Shafer, R. Srikant, A. Arning, and T. Bollinger, “The Quest Data Mining System,” in KDD’96, 1996, pp. 244-249.
17. H. Mannila, H. Toivonen, and I. Verkamo, “Efficient Algorithms for Discovering Association Rules,” in AAAI Workshop on Knowledge Discovery in Databases, 1994, pp. 181-192.
18. J. Han and Y. Fu, “Discovery of multiple-level association rules from large databases,” in 21th Intl. Conf. on VLDB, 1995, pp. 420-431.
19. J. Tomita and G. Kikui, “Interactive Web search by graphical query refinement,” in 10th Intl. WWW Conf, 2001, pp. 190-191.
20. I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler, “Yale: Rapid Prototyping for Complex Data Mining Tasks,” in *Intl Conference on KDD*, 2006, pp. 935-940.
21. T. Zhang, R. Ramakrishnan, and M. Livny, “BIRCH: An Efficient Data Clustering Method for Very Large Databases,” in ACM SIGMOD Intl. Conference on Management of Data, 1996, pp. 103-114.
22. R. White and J. Jose, “A study of topic similarity measures,” in 27th Annual Intl. ACM SIGIR Conf. on Research and Development in Info. Retrieval, 2004, pp. 520-521.
23. F. Lin and C. M. Hsueh, “Knowledge map creation and maintenance for virtual communities of practice,” Information Processing and Management: an International Journal, vol. 42, no. 2, pp. 551-568, 2006.
24. P. N. Tan, M. Steinbachm, and V. Kumar, Introduction to Data Mining. Boston, MA: Addison-Wesley, 2005, pp. 539-547.
25. J. Rennie, “Homepage for 20 Newsgroups Dataset,” [Online]. Available: http://people.csail.mit.edu/jrennie/20Newsgroups/