A Dynamic Programming Approach To Document Clustering Based On Term Sequence Alignment

Muhammad Rafi
Faculty of Computer Science, National University of Computer & Emerging Sciences, Pakistan

Mohammad Shahid Shaikh
Faculty of Electrical Engineering, Habib University, Karachi, Pakistan

Abstract: *Document clustering is unsupervised machine learning technique that, when provided with a large document corpus, automatically sub-divides it into meaningful smaller sub-collections called clusters. Currently, document clustering algorithms use sequence of words (terms) to compactly represent documents and define a similarity function based on the sequences. We believe that the word sequence is vital in determining the contextual information of a document. A frequent sequence, maximal sequence or multi-word sequence cannot alone give good contextual information. These sequences can affect the local similarity computation and can compromise the accuracy of the final clusters. Motivated by this, we propose in this paper a dynamic programming approach to document clustering based on term sequence alignment. There are three main contribution through this research: (i) a document representation model is proposed based on sequence of term used, (ii) a similarity measure is defined that uses term sequence alignment score to assign relatedness for a pair of documents, and (iii) a dynamic programming based hierarchical agglomerative clustering (HAC) algorithm is proposed to cluster the documents. Moreover, the closely related works, (a) Frequent Itemset-based Hierarchical Clustering (FIHC) and (b) Text document clustering based on frequent word meaning sequences (CFWS) are extensively evaluated in comparison of proposed algorithm, on classical text mining datasets. The proposed algorithm significantly improves the quality of the clusters produced and is comparable to state-of-the-art text/document clustering algorithms..*

Keywords: *Document clustering, sequence alignment, dynamic programming, algorithm design.*

1. Introduction

Clustering, an unsupervised machine learning techniques has been successfully applied to a wide range of situations in various domains. The clustering process implicitly learns about the patterns and based on this creates succinct sub groups called clusters. Document clustering is a special case of clustering, where objects are documents. The clustering process for documents has three distinct steps: (i) representation of documents (ii) similarity measure for a pair of documents, and (iii) nature of clustering algorithm. Traditionally, document clustering algorithms mainly uses features like words, phrases, and sequences from the documents to perform clustering. These algorithms generally apply simple features extraction techniques that are mainly based on feature counting and frequency distribution of the features to decide about the relatedness of documents. All these approaches thus, are not able to cater to the meaning behind the text (words). These techniques simply perform clustering independent of the context. We believe that the context of a document in natural languages strongly depends on selection of word sequence for conveying the information. Another view of document clustering problem is that it is an optimization problem. The output of any document clustering algorithm significantly depends on the local similarity functions applied to it. We believe that a dynamic programming approach is well suited for such a problem. A dynamic programming algorithm efficiently evaluates all possible ways to solve a problem and selects the best one. In document clustering, even if one has 10 documents there is large number of possible solutions and the solution with the best evaluation in terms of number of clusters and internal elements (documents) in each cluster is termed the best possible clustering arrangement for such a collection. In this paper, a novel approach that exploits all possible solution space through dynamic programming is presented. The documents are compactly represented as sequence of terms (words) used in the documents. A similarity measure is defined based on common term sequence and sequence alignment. Based on dynamic programming a hierarchical agglomerative clustering (HAC) algorithm is proposed to cluster the documents. The algorithm also ensures to explore the entire solution space. We called this approach dynamic programming approach to document clustering based on term sequence alignment (DPDCSA). The approach is extensively evaluated by performing a series of experiments using standard text mining datasets. The result significantly improves the quality of the clusters produced and is comparable to state-of-the-art text clustering algorithms.

The rest of the paper is organized as follows: Section 2 reviews the related work. In section 3, we describe our document clustering algorithm and its core components. Section 4 presents our experiments. Section 5 discusses the conclusion and future work.

2. State of the art

Clustering as an unsupervised machine learning method, is an effective data mining technique that has been comprehensively studied and extensively applied to a variety of application areas. Details of different data clustering approaches can be found in [1]. Document clustering is a specialized data clustering problem, where the objects are in the form of documents. The objective of the clustering process is to group the documents, which are similar in some sense like type of document, contents of document, etc, into a single group (cluster). The difficult part is to learn from a dataset, actually how many classes of such groups exist in the collection (number of clusters). Document Clustering aims to discover natural grouping among documents in such a way that documents within a cluster are similar (high intra cluster similarity) to one another and are dissimilar to documents in other clusters (low inter cluster similarity). Exploring, analyzing and correctly classifying the unknown natures of data in a document without supervision is a major requirement of document clustering method. Clustering is an effective method for search computing [3]. It offers the possibilities like: grouping similar results [5], comprehending the links between the results [8] and creating the succinct representation and displaying search results. There are two main categories of document clustering (i) Hierarchical and (ii) Partitioning. In partitioning based method the document corpus is partitioned and a flat, non-hierarchal clustering consisting of k clusters are produced. The famous K-means algorithm belongs to this category. The K-means algorithm iteratively refines a randomly selected set of K initial centroids, minimizing the average distance of documents to their closest centroids. A variation to K-mean is called Bisecting K-mean. Bisecting K-mean first selects a cluster to split and then applies K-mean algorithm to get the two resultant clusters. The process iteratively applies till the desired numbers of clusters are obtained. Steinbach's [5] work shows that the bisecting K-mean algorithm outperforms the basic K-mean and agglomerative hierarchical clustering in terms of accuracy and efficiency. Hierarchical clustering method works by grouping data elements into a tree like structure (Hierarchy). There are two variations of Hierarchical clustering (i) Agglomerative and (ii) Divisive. In Agglomerative hierarchical document clustering, each document is assumed to be a single cluster. Iteratively a pair-wise similarity measure is applied to select the two most similar clusters and merges them to reduce the number of clusters. The algorithm works in this bottom-up fashion until all documents fall into a single cluster. In divisive hierarchical clustering, initially all documents are considered to be in a single cluster. The cluster is split into two by using some dissimilarity criteria. These splits continue till each document finally comes into a single cluster. The work proposed in this paper uses hierarchal agglomerative clustering. There are two main objections to agglomerative methods. First, they have low clustering accuracy as the algorithm cannot adjust once the merge step is performed. Second, they also have high computational cost. In our algorithm we address the first objection, by using dynamic programming method. At each step, we perform the merging by analyzing the whole possibility of merge. It further increases the computational cost but, we do not address the computation cost issue in our current research. A recent trend in document clustering is to use frequent itemsets to produce cluster hierarchies. The motivation of this approach comes from transactions based market basket analysis, in which association rules are applied to find transactions which carry a group of items together (itemsets). In document clustering, documents are treated as transactions and the same phenomenon is applied. In FIHC[2], the authors propose a document clustering method that uses frequent itemsets with some prior support level to represent and identify documents. Thus, dimensionalities of the documents are reduced. The algorithm works in four steps: (i) finding global frequent itemsets, (ii) forming initial clusters (iii) forming tree of clusters, and (iv) pruning. The algorithm proposed in this paper is very similar to the work of [6], as we also use sequences but our focus is on the complete sequence that appears in the document. To define similarity we use a sequence alignment technique to incorporate the effect of the entire sequence in the similarity calculation. One more recent work that exploits the frequent sequence is document clustering based on frequent word sequence (CFWS) proposed in [11]. It has three steps: (i) finding the frequent word sequences and collecting the cluster candidates, (ii) merging the cluster candidates based on the k-mismatched concept, and (iii) combining the overlapping clusters with the final clusters. In the first step, the given document collection is scanned to find the set of frequent 2-word sets from the documents, with some user-specified minimum support. Later they build a word-set (WS) for all the frequent words. The document is then represented by only these frequent words and hence the document dimension/size is reduced. Next, the algorithm inserts each compact document into a generalized suffix tree (GST). The cluster candidates are produce by traversing GST in depth-first order, by obtaining the frequent word sequence that appears in a set of documents (candidate cluster). The final step merges the candidates' clusters based on the k-mismatch concepts for a given k. The authors also propose a frequent word meaning sequence based clustering in [11], where they enrich the document representation by using the word-meaning sequence from an online lexical reference system. Our work is different in two aspects (i) we use the entire sequence of terms after preprocessing. Hence our representation of document is quite rich in features, (ii) in place of k-mismatch we have use a sequence alignment method to align the large sequence of words. A relative score is computed based on the maximum match of the

sequence. In the study of [6], the authors claim that the FIHC produces low-quality clusters on several datasets when the algorithm runs with best support thresholds. Similarly, CFWS and CFWMS also produce inconsistent results with best thresholds selected [7]. The effect of large sequence can be poor similarity function; this effect is mitigated by: (ii) a sequence alignment score. The sequence alignment is very similar to k-mismatch. We have selected three algorithms for comparison with our proposed method. These are (i) Bisecting K-mean (BKM), (ii) FICH and (iii) CFWS.

3. Proposed Approach (DPDCSA)

The proposed approach for document clustering is named as "Dynamic Programming approach to Document Clustering based on sequence alignment" (DPDCSA).

3.1 Document Clustering

The problem of document clustering tries to subdivide a collection of documents into smaller, more meaningful and manageable collections (clusters). The unsupervised nature of this sub-division implicitly tries to find the natural grouping characteristics of document, this is very challenging. The clustering process for documents has three distinct steps: (i) representation of document (ii) similarity measure for a pair of document and (iii) nature of clustering algorithm. In this approach, we define a new representation of document based of term-sequence. A document is parsed and terms are extracted after performing the conventional preprocessing of document/text. The sequence of terms that appear in any document forms a representation for that document. For illustration purpose, consider the following set of over simplified documents.

D_1: I love to play cricket. I hate hockey.
D_2: Pakistani boys love cricket and hockey.
D_3: Hockey is the national game of Pakistan. Cricket is mainly popular and loved by majority.

The preprocessing of the documents, like removing stop words gives the following compact representation.

Ds_1: {love, play, cricket, hate, hockey}.
Ds_2: {Pakistan, boy, love, cricket, hockey}
Ds_3: {hockey, national, game, pakistan, cricket, mainly, popular, loved, majority}

3.2 Dynamic Programming

Dynamic programming is a technique of problem solving in which the original problem is break down into simpler sub-problem, if the original problem has implicit characteristics of overlapping sub-problem and optimal substructures. Dynamic programming based solution to the problem that exhibits these two characteristics, guarantees to give optimal solution. A dynamic programming based solution examines all possible solution intelligently to search the possible optimal (best) solution.

3.3 Sequence Alignment and Document Similarity

Sequence Alignment is an important phenomenon in bioinformatics. The process of sequence alignment is generally used in DNA, RNA, or protein to identify regions of similarity that may be a consequence of functional, structural, or evolutionary relationships between them. In document clustering we define a sequence alignment of terms (words) as below:

A document Di is represented as a sequence of terms X of length n, is defined as a linear succession of strings of different length from a finite alphabet set L. Consider the representation example document D_2 and D_3.

Ds_2: {Pakistan, boy, love, cricket, hockey}
Ds_3: {hockey, national, game, pakistan, cricket, mainly, popular, loved, majority}

In this form documents D_2 and D_3 are called document sequence Ds_2 and Ds_3 respectively. The basic sequence alignment based similarity of documents, is based on the three terms. These are defined as below:

3.4 Core-Term-Pairs (CTP)

In any pair of documents, D_i and D_j the core-term-pairs are common terms in the two sequences. Consider the two document sequence as a running example:

Ds_1: {***love***, play, ***cricket***, hate, ***hockey***}.
Ds_2: {Pakistan, boy, ***love***, ***cricket***, ***hockey***}

There are three core-term-pairs in the above example. These are { (love $^{1}_{ds1}$, love $^{3}_{ds2}$), (cricket $^{3}_{ds1}$, Cricket $^{4}_{ds2}$), (Hockey $^{5}_{ds1}$, Hockey $^{5}_{ds2}$)}. The pair (love $^{1}_{ds1}$, love $^{3}_{ds2}$) means that the term 1 of document sequence 1, is paired with term 3 of document sequence 2, which are the sequence representation of documents D1 and D2, thus, there are three core-term pairs. A pair of documents is more similar if they contain more core-term-pairs.

3.5 Core-Term-Alignment-Score (CTAS)

The numbers of core-term-pairs define the contribution of these core terms to the similarity of the two documents. The semantics of a pair of documents are closer, if the core-term-pairs are more close to each other. As an example let the position of current CTP in the two sequences be (x,y) and the position of the previous core term match is (x', y'). The condition that x' < x and y'< y must be satisfied as the expression $(x - x')(y - y')$, we compute the CTAS with the following equation:

$$CTAS(x,y) = \frac{1}{(x - x')(y - y')}$$

In our example,

CTP	(love $^{1}_{ds1}$, love $^{3}_{ds2}$)	(cricket $^{3}_{ds1}$, Cricket $^{4}_{ds2}$)	(Hockey $^{5}_{ds1}$, Hockey $^{5}_{ds2}$)
Prev. Match	nil	(love $^{1}_{ds1}$, love $^{3}_{ds2}$)	(cricket $^{3}_{ds1}$, Cricket $^{4}_{ds2}$)
CTAS	1/3	1/2	1/2

3.6 Sequence-Alignment-Score (SCS)

The numbers of core-term-pairs define the contribution of these core terms to the similarity of the two documents. The sequence alignment score (SCS) is defined as the sum of the CTAS. If E is defined the set of core-terms-pair from the two documents. Then SCS can be computed by the following equation:

$$SCS = \sum_{(x,y) \in E} CTAS(x,y)$$

The sequence alignment score (SCS) for the running example is calculated as
SCS= 1/3+ 1/2+1/2 ≈ 4/3 = 1.33

3.7 Sequence Similarity Measure (SSM)

The document clustering algorithm is very sensitive to similarity measure used to compute the relatedness between two pairs of documents. The sequence similarity measures (SSM) define with the following equation:

$$Sim(Di, Dj) = \frac{Dsi \cap Dsj}{Dsi \cup Dsj} \equiv \frac{SCS}{(|Dsi \cup Dsj|)}$$

It is quite obvious from the above equation that the similarity is symmetrical that is $Sim(Di, Dj) = Sim(Dj, Di)$. The scaling factor $(|Dsi \cup Dsj|)$ also ensures that the similarity values computed between 0 and 1.
In our running example, we will get the following values:
SCS=1.33 and $Dsi \cup Dsj$ = { love, play, cricket, hate, hockey, Pakistan, boy}
Hence, $Sim(Di, Dj) = \frac{1.33}{7}$ =0.19

Algorithm: Sequence-Similarity-Measure (SSM)
Input: Documents Sequences Dsi, Dsj with terms counts ***m*** and ***n*** respectively
Output: Similarity between Di and Dj
Step 1: integers x',y', sum=0;
Step2: for (int curX=0; currX< m; currX++){
Step3: for (int curY=0; currY< n; currY++){
Step4: if (Dj.curY is not processed) and (Dj.curY==Di.curX)
Look for nearest previous Core-Term-Pair from (curX,curY) as (X',Y')
SCS= SCS + CTAS(x',y')
marked Dj.curY as processed
sum++; }}
Step5: return SCS/sum;

3.8 Clustering Algorithm
Our proposed algorithm for document clustering uses dynamic programming to search the possible solution space for the optimal (best) cluster assignments depending on the sequence of terms (topic-words). We modify a bottom-up agglomerative hierarchical clustering algorithm to converge quickly in search space using dynamic programming. Initially each document is assigned to its own cluster. The sequence of the topics appears in this document is used to compactly represent this document. Next, we repeatedly select the clusters with large topics sequence, and merge them to produce large clusters. There we used a criterion function that's tries to maximize its value. The criterion function (Cf) is given below:

$$Maximize\ Cf = \sum_{r=1}^{k} \frac{1}{nr} \sum_{Di,Dj \in Cr} SSM(Di, Dj)$$

In the above function nr is the number of sequence in Cr and k is the number of clusters.
Algorithm: DPDCSA
Input: Document collection represented in Topic-Sequence, the number of clusters k
Output: set of Clusters (C1, C2, ..Ck)
Step1: for each document Di in the database D
Clusters.Ci = Di;
Step2: int TotalClusters=|D|
Step3: double Cf=0.0;
Step 4: while TotalCluster > k do {
int m=1,n=1;
for each Ci and Cj from Clusters {

$$Cf' = Cf - \sum_{r=i,j} \frac{1}{nr} \sum_{Di,Dj \in Cr} SSM(Di, Dj) + \frac{1}{ni+nj} \sum_{Di,Dj \in Ci \cup Cj} SSM(Di, Dj)$$

if (Cf' > Cf) { Cf=Cf'; m=i;n=j;}
} // end for
Cnew= merge(Cm,Cn);
} //end while
Step5: return Clusters;

4. Experiments

4.1 Experimental Setup
The algorithmic method proposed in this paper, is implemented and tested through a series of experiments. The hardware setup comprises of a Dell Vostro Notebook, with Intel Core2duo processor, 2GB of RAM with 200GB of hard disk storage. The software setups include the suggested approach implemented in C# programming language. We have used Cluto Clustering toolkit [4] generating solution for Bisecting K-means. We have implemented CFWS as per the algorithm suggested in [9] in C# as well. The FIHC is also implemented as per the algorithm in [2].

4.2 Datasets

The three standard document datasets are used to compare the quality of our clustering method. These three data sets are selected mainly due to the fact that most researchers whom work is related to this study have used the same datasets to report their results and comparisons. These datasets are:

Reuters: The Reuters-21578, test collection of Distribution 1.0 is used. The collection appeared in Reuter's newswire in the year 1987. The collection consists of 22 data files, an SGML DTD file describing the format of the available data, and six files describing the categories used to index data. The collection is available at http://www.daviddlewis.com/resources/testcollections/reuters21578/

NEWS20: It is also a popular data set among text mining community; it's mainly used for text classification and clustering measure for machine learning techniques. The data set consists of approximately 20,000 newsgroup documents, partitioned in 20 different classes. The data set is available at http://people.csail.mit.edu/jrennie/20Newsgroups/

OHSUMED: The OHSUMED collection of the 1987-1991 abstract from 270 journals. It consists of over 348,566 references from the MEDLINE database, which is a database of medical literature maintained by the National Library of Medicine (NLM). Most of the references have abstracts and all have associated MeSH (Medical Subject Headings) indexing terms, with some of the MeSH terms marked as primary. The data set is available at http://davis.wpi.edu/xmdv/datasets/ohsumed.html

We have selected subsets of data from these datasets. We have selected Re1, Re2, and Re3 are selected from Reuters-21578. The subsets N20a, N20b, and N20c are selected from NEWS20. The dataset OHS1 and OHS2 are selected from OHSUMED. Each document of the test data sets has been pre-classified into one or more classes. This information is hidden during the clustering processes and used to evaluate the clustering quality of each clustering algorithm in term of accuracy. The summary of the selected sub-sets of data for the experiments are given in the table 1.

4.3 Document Preprocessing

A document "cleaning" process is used for all data sets, before performing actual document clustering. The first step of the process stripped off all the non-word tokens, next the text is parsed into words. The second step, identify and removed all stop words by using a standard stop word list.

4.4 Evaluation Metrics

We justify the effectiveness of our proposed method by using standard cluster quality measures like F-Measure and purity.

F-Measure

Dataset	#of doc.	#classes	# of unique terms	# of total terms
Re1	350	8	3218	23876
Re2	700	10	4988	112301
Re3	1400	15	5646	86743
N20a	400	8	6570	87561
N20b	800	10	9875	112300
N20c	1600	18	16765	150431
OHS1	400	8	8700	65787
OHS2	800	16	12436	147652

Table 1: Summary of Selected Datasets

F-Measure is commonly used measure in evaluating the effectiveness of clustering and classification algorithms. The F-measure uses a combination of *precision* and *recall* values of clusters. Assume that the document base (D) comprises of N documents, the clustering algorithm produces $C=\{C_1,C_2,..C_k\}$ clusters. The document base has actual correct clusters $C^*=\{C^*_1, C^*_2,.. C^*_l\}$. Then, the *recall* of cluster *j* with respect to class *i*, *rec(i,j)* is define as $| C_j \cap C^*_i| / | C^*_i|$. The *precision* of cluster *j* with respect to class *i*, *prec(i,j)* is defined as $| C_j \cap C^*_i| / | C_j|$. F-Measure combines both precision and recall with the following formula:

$$F(i,j) = \frac{2 * prec(i,j) * rec(i,j)}{prec(i,j) + rec(i,j)}$$

The F-measure for overall quality of cluster set C is defined by the following formula:

$$F = \sum_{i=1}^{k} \frac{|C_i|}{N} * \max_{i=1,2,..k} \{F(i,j)\}$$

Purity

Purity can be defined as the maximal precision value for each class j. The cluster purity indicates the percentage of the dominant class members in the given cluster. The overall purity of the cluster C, can be computed as the weighted average purity by the following formula:

$$Purity = \sum_{j=1}^{k} \frac{|C_j|}{N} * \max_{i=1,2,..k} \{Prec(i,j)\}$$

5. Results and Discussion

We compare our proposed algorithm DPDCSA with three recently proposed algorithms, (i) Bisecting K-means, (ii) FIHC and (iii) CFWS. The F-measure is calculated on the eight selected datasets. Bisecting K-mean is performed by using Cluto[9]. We noticed that Bisecting K-means performance is consistent in all datasets [10]. In case of FIHC, we executed the algorithm on each dataset several times with different support thresholds; we reported the average of the best three results obtained from these runs. We noticed that each datasets produce best results on some different thresholds level for example: Reuters dataset the support level was 5%, while NEWS20 datasets the support level was 3% and OHSUMED it was 2%. In case of CFWS,

we used the same support level as determine by FIHC execution. There were only two cases, on NEWS20 datasets, where CFWS performance is higher than FIHC, which was on N20a and N20b. Table 2 present the F-measure for all the datasets in all three algorithms.

	BKM	FIHC	CFWS	DPDCSA
Re1	0.451	0.541	0.521	0.588
Re2	0.482	0.582	0.568	0.623
Re3	0.499	0.612	0.632	0.668
N20a	0.472	0.566	0.644	0.635
N20b	0.522	0.589	0.687	0.689
N20c	0.541	0.623	0.687	0.712
OHS1	0.476	0.566	0.633	0.698
OHS2	0.523	0.566	0.647	0.698

Table 2: F-measure on different datasets

The same result in graphical form.

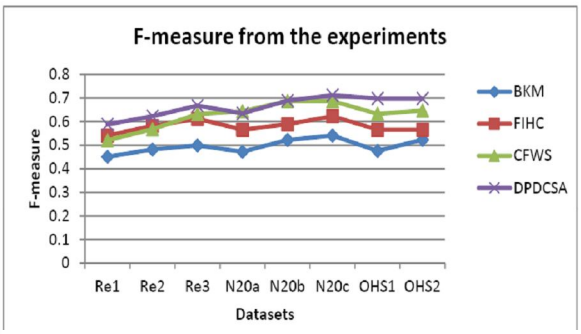

Fig 1: Graphical results of F-Measure

The second evaluation is purity for our experiments. Table 3, shows the purity values for different algorithms on the selected datasets. It is quite clear from the table that DPDCSA outperforms all the three algorithms on purity. The higher purity value is an indication of producing more homogenous clusters.

	BKM	FIHC	CFWS	DPDCSA
Re1	0.666	0.497	0.751	0.791
Re2	0.712	0.534	0.731	0.81
Re3	0.742	0.564	0.719	0.831
N20a	0.681	0.633	0.733	0.792
N20b	0.713	0.681	0.733	0.823
N20c	0.741	0.699	0.745	0.842
OHS1	0.541	0.56	0.719	0.773
OHS2	0.611	0.661	0.722	0.802

Table 3: Result of purity on selected datasets

The same results on graphical format.

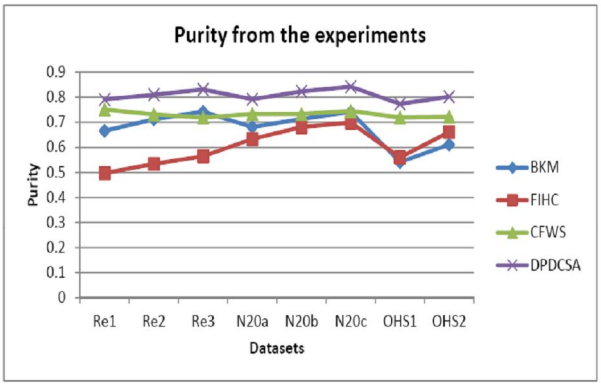

Fig 2: Purity values from the experiments

6. Conclusion & Future Work

We introduced a dynamic programming approach to document clustering based on sequence alignment (DPDCSA). The documents are represented by the term sequences used in them. The sequence order is preserved. Unlike the representation based on frequent sequence, maximal frequent sequences, closed sequence or close interesting sequences, dimensionality is not drastically reduce. We believe that this representation captures more semantics of the content. We define a similarity measure by using sequence alignment of the common term between a pair of document. This incorporate both the lexical and positional semantic in the similarity computation. The dynamic programming based hierarchical agglomerative clustering algorithm is defined that utilizes sequence alignment similarity score to produce the final clusters. The proposed algorithm searches the entire solution space for the best overall criteria value; hence it optimizes the accuracy of the clusters. We compare the proposed algorithms with two of the quite similar work recently proposed (i) Frequent Itemset-based Hierarchical Clustering (FIHC) [B.C.M. Fung, K. Wang, M. Ester, Hierarchical document clustering using frequent itemsets, in: Proceedings of SIAM International Conference on Data Mining, 2003] and (ii) Text document clustering based on frequent word meaning sequences (CFWS)[Y.J. Li et al., Text document clustering based on frequent word meaning sequences, Data & Knowledge Engineering 64 (2008) 381–404]. Our proposed algorithm DPDCSA, clearly out performs these on three classical text mining datasets. In future, there are several directions to extend this research work. The sequence representation can be change to multi-sequence, the alignment technique can also be altered and its effect on clustering can be an interesting study. The DPDCSA nature of dynamic programming can be study further for optimization and recomputed/ memorization based execution speedups.

References

[1] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data Clustering: a review," ACM Computing Survey, pp. 264-323, 1999.

[2] B. Fung, K. Wang, and M. Ester, “Hierarchical document clustering using frequent itemsets”, In Proc. SIAM International Conference on Data Mining, 2003, pp. 59-70.

[3] Campi, A. and Ronchi, S., "The Role of Clustering in Search Computing ," in 20th International Workshop on Databases and Expert Systems Application , Linz, Austria, 2009, pp. 432-436.

[4] Cluto http://glaros.dtc.umn.edu/gkhome/views/cluto

[5] Cutting, D. R., Karger, D. R., Pedersen, J. O., and Tukey, J. W., "Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections,"

in Fifteenth Annual International ACM SIGIR Conference, June 1992, pp. 318-329.

[6] Hassan H. Malik and John R. Kender. High quality, efficient hierarchical document clustering using closed interesting itemsets. ICDM, 0:991-996, 2006.
[7] Rafi, M.Maujood, M.M.Fazal, S.M.Ali, " A Comparision of Two Suffix Tree Based Document Clustering Algorithm", in Proc. of International Conference on Information and Emerging Technologies, Karachi, Pakistan, 2010
[8] M. A. Hearst, and J. O. Pedersen, "Reexamining the cluster hypothesis: scatter/gather on retrieval results," in 19th annual international ACM SIGIR conference on Research and development in information retrieval, Zurich, Switzerland , 1996, pp. 74-84.
[9] Muhammad Rafi, Shahid M Shaikh and Amir Farooq. "Document Clustering based on Topic Maps". International Journal of Computer Applications 12(1):32–36, December 2010.
[10] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. KDD Workshop on Text Mining'00, 2000.
[11] Y.J. Li, S.M. Chung, J.D. Holt, "Text document clustering based on frequent word meaning sequences," Data & Knowledge Engineering, vol. 64, pp. 381–404, 2008.

Muhammad Rafi received his BS(Computer Science) and MS(Computer Science) with a gold medal in 1996 and 2000 respectively. He is currently a final year PhD student in Computer Science at NUCES-FAST, Karachi Campus. His research encompasses data/text mining, algorithm development, machine learning and theoretical computation models. He is a member of IEEE and ACM.

Mohammad Shahid Shaikh received the BE degree from Mehran University of Engineering and Technology, Pakistan, in 1986, the MS from Michigan State Univeristy in 1989 and the PhD degree from McGill University, Montreal, in 2004, all in Electrical Engineering. He is currently associate professor of Electrical Engineering at Habib University, Karachi, Pakistan.