Document Clustering Approach Using Internal Criterion Function

Ch.Sivasankar
Department of Computer Science and Engineering
Annamacharya Institute of Technology &Sciences
TIRUPATI, A.P-517520

D.Vivekananda Reddy
Department of Computer Science and Engineering
SV University
SVUCE
TIRUPATI, A.P-517502

Abstract:-Document clustering is the act of collection of similar documents into bins. Fast and high quality document clustering is an important task in organizing information, search engine results obtaining results from user query, enhancing web crawling and information retrieval. With the large amount of data available and with a goal of creating good quality clusters, a variety of algorithms have been developed having quality-complexity trade offs. Among these, some algorithms seek to minimize the computational complexity using certain criterion functions which are defined for whole set of clustering solution. This paper proposes a novel document clustering algorithm based on internal criterion function. Commonly used partitioning clustering algorithms (e.g. k-means) have some drawbacks as they suffer from local optimum solutions and creation of empty clusters as a clustering solution. The proposed algorithm usually doesn't suffer from these problems and converge to a global optimum. Further its performance enhances with the increase in the document number of clusters. The proposed algorithm has been verified against three different datasets for four different values of k (required number of clusters). Further CLUTO tool has been used to improve the performance of criterion function for document clustering.

General Terms
K-means clustering, hierarchical clustering, repeated bisection.

Keywords
Document clustering, quality, dimensional, CLUTO.

I. INTRODUCTION

Developing an efficient and accurate clustering algorithm has been one of the most favorite areas of research in various scientific fields. Various algorithms have been developed over a period of years [2, 3, 4, 5]. These algorithms can be broadly classified into agglomerative [6, 7, 8] or partitioning [9] approaches based on the methodology used or into hierarchical or non-hierarchical solutions based on the structure of solution obtained.Hierarchical solutions are those which are in the form of a tree called dendograms [15], which can be obtained by using agglomerative algorithms, in which, first each object is assigned to its own cluster and then pair of clusters are repeatedly joined until a certain stopping condition is not satisfied. On the other hand , partitioning algorithms such as k-means [5], k-medoids [5], graph-partioning-based [5] consider whole data as a single cluster and then find clustering solution by bisecting or partitioning it into number of predetermined classes. However, a repeated application of partitioning application can give a hierarchical clustering solution.

There always involves tradeoffs between a clustering solution quality and complexity of algorithm. Various researchers have shown that partitioning algorithms in terms of clustering quality are inferior in comparison to agglomerative algorithms [10]. However, for large document datasets they perform better because of small complexity involved [10, 11].

Partitioning algorithms work using a particular criterion function with the prime aim to optimize it, which determines the quality of clustering solution involved. In [12, 13] seven criterion functions are described categorized into internal, external and hybrid criterion functions. The Best way to optimize these criterion functions in partitioning algorithmic approach is to use greedy approach as in k-means. However the solution obtained may be sub-optimal because many a times these algorithms converge to a local-minima or maxima. Probability of getting good quality clusters depends on the initial clustering solution [1]. We have used an internal criterion function and proposed a novel algorithm for initial clustering based on partitioning clustering algorithm. In particular we have compared our approach with the approach described in [1] and implementation results show that our approach performs better then the above method. Text mining could be broadly defined as a knowledge-intensive process in which a user interacts with a document collection over time by using a suite of analysis tools. In a manner analogous to data mining, text mining seeks to extract useful information from data sources through the identification and exploration of interesting patterns. In the case of text mining, however, the data sources are document collections, and interesting patterns are found not among formalized database records but in the unstructured textual data in the documents in these collections. Text mining derives much of its inspiration and direction from seminal research on data mining. Therefore, it is not surprising to find that text mining and data mining systems evince many high-level architectural similarities. Further, text mining adopts many of the specific types of patterns in its core knowledge discovery operations that were first introduced and vetted in data mining research. Because data mining assumes that data have already been stored in a structured format, much of its preprocessing focus falls on two critical tasks: Scrubbing and normalizing data and creating extensive numbers of table joins. In contrast, for text mining systems, preprocessing operations center on the identification and extraction of representative features for natural language documents. These preprocessing operations are responsible for transforming unstructured data stored in document collections into a more explicitly structured intermediate format, which is a concern that is not relevant for most data mining systems. Moreover, because of the centrality of natural language text to its mission, text mining also draws on advances made in other computer science disciplines concerned with the handling of natural language; most notably, text mining exploits techniques and methodologies from the areas of information retrieval, information extraction, and corpus-based computational linguistics.

Physicians and sports analyst's interpretations of images, signals, or any other data, are written as unstructured free-text reports or documents. Such documents are very difficult to standardize and thus difficult to mine, even specialists from the same discipline cannot agree on unambiguous terms to be used. Developing methods or techniques to organizing large amount of unstructured documents into a small number of meaningful clusters will help users to find what they are looking for, extract meaningful information and discovering trends and patterns hidden within these documents more effectively, because dealing with only the cluster that will contain relevant documents should improve effectiveness and efficiency. The produced clusters contain groups of documents that are more similar to each other than to the members of any other group. Therefore, the goal of finding high-quality document clustering algorithms is to determine a set of clusters such that inter-cluster similarity is minimized and intra-cluster similarity is maximized. Since further knowledge extraction and data mining will be applied to the produced clusters, achieving high-quality clustering solution is important. Document clustering has been investigated for usage in different areas such as browsing collections of documents, improving the precision and recall in information retrieval systems, automatically generating hierarchical clusters of documents etc. There are two clustering algorithms approaches based on the underlying methodology: agglomerative and partitional approaches. There have been many conclusions derived in different studies that investigated the clustering performance of agglomerative and partitional approaches. The partitional clustering algorithms are well suited for clustering large documents datasets due to their relatively low computational requirements according to study conducted In terms of clustering quality, work reported concluded that the partitional algorithms are actually inferior and less effective than their agglomerative counterparts. Using datasets from TREC and Reuters, Larsen and Aone observed that agglomerative clustering outperformed various partitional clustering algorithms. None of these studies addressed the effect of the criterion functions. Criterion functions optimize the entire clustering process for both approaches. A recent study reported by Zhao and Kapyris investigated the effect of the criterion functions to the problem of partitionally clustering documents and the results showed that different criterion functions lead to substantially different results. Another study reported in investigated the effect of the criterion functions to partitional and agglomerative clustering algorithms using various document datasets obtained from various sources and their results showed that partitional algorithms always led to better clustering results than agglomerative algorithms

The information age has made it easy to store large amounts of data. The proliferation of documents available on the World Wide Web, on corporate intranets, on news wires, and elsewhere is overwhelming. However, although the amount of data available to us is constantly increasing, the ability to absorb and process this information remains constant. Search engines only exacerbate the problem by making more and more documents available in a matter of a few key strokes.

Section 1 is all about introduction to Text Mining and its features, document clustering. It also gives the organization of the thesis. The Literature on document clustering and criterion functions is reviewed in Section 2, which describes various algorithms and discusses the necessary properties. Document clustering using criterion function problem definition is discussed in Section 3. The model of the algorithm is discussed in Section and word processing 4. An algorithm, implementation and design are presented in Section 5. The performance of the algorithm is presented in Chapter 6. Conclusions and scope for future enhancements are discussed in Section 7.

II. TERMINALOGY

In this paper documents have been represented using a vector-space model [14]. This model visualizes each document, d as a vector in the term-space or more in more precise way each document d is represented by a term-frequency (T-F) vector.

d , 1, 2, where denotes the frequency of the th term in the document. In particular we have used a term-inverse document frequency (tf-idf) term weighing model [14]. This model works better when some terms appearing more frequently in documents having little discrimination power need to be de-emphasized. Value of idf is given by log/d), where N is the total number of documents and d is the number of documents that contain the term.
$d - d$ = (1 log(N/d 1), 2 log(N/d 2),................, log(N/d)).

The documents are of varying length, the document vectors are normalized thus rendering them of unit length (|$d - d$ |=1). In order to compare the document vectors, certain similarity measures have been proposed. One of them is cosine function [14] as follows

Cos (d , d) = d d ||d || ||d ||……………………..(1)

Where, d , d are the two documents under consideration, || d || and ||d || are the lengths of vector d and d respectively. This formula , owing to the fact that d and d are normalized vectors ,converges into

Cos (d , d) = d d ……………………………………………(2)

The other measure is based on Euclidean distance, given by Dis (d , d) = (d −d) (d −d) = || d −d ||. Let A be the set of document vectors, the centroid vector is defined to be = | | where, represents composite vector given by $d_{d\in}$.

III. DOCUMENT CCLUSTERING

Clustering is an unsupervised machine learning technique. Given a set of documents, we define clustering as a technique to group similar documents together without the prior knowledge of group definition. Thus, we are interested in finding k smaller subsets (i = 1, 2,.........k) of such that documents in same set are more similar to each other while documents in different sets are more dissimilar. Moreover, our aim is to find the clustering solution in the context of internal criterion function.

A. Internal Criterion Function

Internal criterion functions account for finding clustering solution by optimizing a criterion function defined over documents which are in same set only and doesn't consider the effect of documents in different sets.
The criterion function we have chosen for our study attempts to maximize the similarity of a document within a cluster with its cluster centroid [11]. Mathematically it is expressed as

Maximize T = (d ,) d € =1………………………………………..(3)

Where, d is the document and is the centroid of the cluster.

3.1. *ALGORITHM DESCRIPTION*

Our algorithm is basically a greedy one, unlike other partitioning algorithm (e.g. k-means) it generally does not converge to a local minimum.
Our algorithm consists of mainly two phases (i) initial clustering (ii) refinement.

A. Initial clustering

This phase consists of determining initial clustering solution which is further refined in refinement phase, with the assumption
In this phase of algorithm, our aim is to select K documents, hereafter called seeds, which will be used as initial centroid of K clusters required.

We select the document which has minimum sum of squared distances from the previously selected documents. In the process we get the document having largest minimum distance from previously selected documents, i.e., document which is not in the neighborhood of currently present documents.
Let at some time we have m documents in the selected list, we check the sum S =$\sum_{i=1,}^{k}$ ((d ,))^2 for all documents a in set A, where set A contains the documents having largest sum of distances from previously selected m documents, and finally the document having minimum value of S, is selected as the (m+1)th document. We continue this operation until we have K documents in the selected list.

1) *Algorithm:*

```
Step1: DIST □□adjacency matrix of document vectors
Step2:RP □□regulating parameter
Step3: LIST □□set of document vectors
Step4:N □□number of document vectors

Step5:K □□number of clusters required
Step6:ARR_SEEDS □□list of seeds initially empty
Step7:Add a randomly selected document to ARR_SEEDS
Step8: Add to ARR_SEEDS a new document farthest from the residing document in ARR_SEEDS
Step9: Repeat steps 10 to 13 while ARR_SEEDS has less than K elements
Step10: STORE □□set of pair ( sum of distances of all current seeds from each document, document ID)
```

Step11: Add in STORE the pair(sum of distances of all current seeds from each document, document ID)
Step12: Repeat Step 13 R times
Step13: Add to ARR_SEEDS the document having least sum of squared distances from available seeds
Step14: R epeat 15 and 16 for all remaining documents
Step15: Select a document
Step16: Assign selected document to the cluster corresponding to its nearest seed

2) *Description:* The Algorithm begins with putting up of a randomly selected document into an empty list of seeds named ARR_SEEDS. We define a seed as a document which represents a cluster. Thus we aim to choose K seeds each representing a single cluster. The most distant document from the formerly selected seed is again inserted into ARR_SEEDS. After the selection of two initial seeds, others are to be selected through an iterative process where in each iteration we put all the documents in descending order of their sum of distance from the currently residing seeds in ARR_SEEDS and then from the ordered list we take top R (regulating variable which is to be decided by the total number of documents, the distribution of the clusters in K-dimensional space and the total number of clusters K) documents to find the document having minimum sum of squared distances from the currently residing seeds in the list, the document thus found is added immediately into ARR_SEEDS and more iterations follow until number of seeds reach K. The variable R is a regulating variable which is to be decided by the total number of documents, the distribution of the clusters in K-dimensional space and the total number of clusters K.

Now we have K seeds in ARR_SEEDS each representing a cluster. For the remaining N-K documents, each document is assigned to the cluster corresponding to its nearest seed.

B. Refinement

The refinement phase consists of many iterations. In each iteration all the documents are visited in random order, a document d is selected from a cluster and it is moved to other k-1 clusters so as to optimize the value of criterion function. If a move leads to an improvement in the criterion function value then d is moved to that cluster. A soon as all the documents are visited an iteration ends. If in an iteration there are no documents remaining, such that their movement leads to improvement in the criterion function, the refinement phase ends.

1) Algorithm:
Step1: S□□Set of clusters obtained from initial clustering
Step2: Repeat steps 3 to 9 until even a single document moved between clusters
Step3: Unmark all documents
Step4: Repeat steps 5 to 9 while each document is not marked
Step5: Select a random document X from S
Step6: If X is not marked , perform Steps 7 to 9
Step7: Mark X
Step8: Search cluster C in T in which X lies
Step9: Move X to any cluster other than C by which the overall criterion function value of S goes down. If no such cluster exists don't move X.

IV. WORD SENSE DISAMBIGUATION

The problem of finding the correct sense of a word in a context is called word sense disamiguation (WSD) [12]. There are two general types of methods for sense determination [10]. Some methods used the definition of each sense in a dictionary. In WordNet, the definition is the gloss assigned to each synset. Other approaches used the semantic relatedness in an existing semantic network. Lesk proposed to disambiguate the sense of a polysemous word based on its context words and the definition of all senses in a dictionary [19]. For each sense of a word, a lexicon definition can be found in a dictionary. All the words occurring in this definition compose the sense bag for this sense. All the words occurring in the definitions of the senses of the context words compose the context bag for this word.The context bag of a word will be compared to each sense bag of all candidate senses. The sense with the maximum match will be selected. Fragos, Maistros, and Skourlas [10] proposed to improve Lesk's method by enriching the sense bag and the context bag. For each word in the sense bag, all the words that have a "hypernymy/hyponymy" relationship with it are also identified. The definitions of these newly identified words are incorporated into the sense bag. For the context words, all the words that have a "hypernymy" relationship with the current context word and their definitions are added into the context bag.Gomes et al. [12] present a word sense disambiguation method based on the semantic distance among the context words in WordNet. Given a set of context words, {*W1, W2, …Wn*}, from WordNet, we can get a set of senses for each word. Suppose the number of senses for word *i* is *si*, then we get a set of senses {*S(1, 1), S(1, 2), …, S(1, s1), S(2, 1), S(2, 2), …,S(2, s2), ….., S(n, 1), S(n, 2), ….S(n, sn)*}. With different measures, we can get the semantic distance between two senses, *SemanticDist(S(i, j), S(k,l))*. The shortest semantic distance between a sense *S(i, j)* and a word *Wk* is: $ShortestDist(S(i,j),Wk) = Min_{1<l<sl}\{SemanticDist(S(i, j), S(k, l))\}$ For this

sense, its synset score is defined as the sum of all shortest semantic distances between this sense and all the other words. The synset score reflects the context semantic distance between this sense and all context words.

$$SynsetScore(S(i,j))=\sum_{k=1}^{n} ShortestDist(S_{ij}W_k) \quad \ldots\ldots\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(4)$$

Then for each word, the sense which has the minimum synset score will be selected as its sense. $Sense(W_k) = Min1<l<sl\{SynsetScore(S(k,l))\}$ Sussna [5] proposed a similar disambiguation method based on semantic distance. Instead of finding the shortest distance between one sense and one word, Sussna's method just uses all senses and tries to find a combination of candidate senses to achieve the minimum total pairwise semantic distance among the selected senses. Li, Szpakowicz, and Matwin [10] proposed eight heuristic rules for sense disambiguation based on WordNet semantic relatedness. The basic idea in this method is try to find verb-noun pairs in the context with semantic relatedness in WordNet. Those related verb-noun pairs were used to determine the sense for each other. Ramakrishnanan and Bhattacharyya [7] also proposed to use the synsets in WordNet to replace the original word for text representation. But they are using a soft sense disambiguation method in which each word will be assigned several senses instead of a single sense. Three algorithms, hubs and authorities, page ranking, and Bayesian inferencing, were described to rank the candidate synsets.

4.1 SEMANTIC RELATEDNESS MEASURE

A semantic relatedness measure is a criterion to scale the relatedness of two senses in a semantic network. It is also called semantic distance or semantic similarity in the literature. In a lot of word sense disambiguation algorithms, a semantic relatedness measure is a very important factor for the performance. Budanitsky presents a comprehensive overview of various semantic relatedness measures [5, 6]. Jiang and Conrath classify those methods into two categories, edge-based methods and node-based (information content-based) methods [17]. Edgebased methods attempt to measure the distance between two senses according to the length of the path between them in the semantic networks. The simplest method is to count the number of edges or nodes between them. Some others include Hso method [15], Lch method [18], Sussna's method [5], and Wup method [8]. Node-based methods measure the distance between two senses according to the statistical information contained in the nodes within the semantic network. There are many different methods used to calculate the information based on a corpus. Some generally used node-based methods include Res method [8, 9] and Lin method [8]. Jcn method is a combined method which considers both edge and node information [17]. Some other methods include Banerjee and Pedersen's method [1, 5], and Patwardhan's method [4]. WordNet::Similarity is a perl software package which consists of several submodules to implement different semantic relatedness measures [6].

1.2 TERM MUTUAL INFORMATION

In this section we focus om mining the term mutual information with the aid of conceptual background knowledge given by ontologies(eg.,WordNet),statistical methods and human assessments.some researchers put their research on learning or extracting ontologies from text[7],[8],[9],[10]. In the linguistic preview they have proved that some relationship exist between the terms so we had better utilize them to express our document vector space rather than only the traditional term based VSM.

In order to find tern mutual information between terms first of all we exploit the background knowledge which is given through an ontology source:WordNet. WordNet[14] is lexical database in which terms are organized in so-called synsets containing synonyms and thus representing a specific meaning of a given term. We combine the background knowledge into the traditional term-based VSM and modifying the term vectors accordingly with the following method.

For each term $\boldsymbol{t_{i1}}$, we check whether it is semantically related to each other term $\boldsymbol{t_{i2}}$ with the WordNet. We use $\boldsymbol{\delta_{i1,i2}}$ to indicate semantic information between two terms. If $\boldsymbol{t_{i2}}$ appears in the synset of $\boldsymbol{t_{i1}}$(here synonyms and hypernym synsets are considered), $\boldsymbol{\delta_{i1,i2}}$ will be treated in a same level for different $\boldsymbol{t_{i1}}$ and $\boldsymbol{t_{i2}}$ otherwise $\boldsymbol{\delta_{i1,i2}}$ will be set zero. With $\boldsymbol{\delta_{\mathrm{i1,i2}}}$ the weight $\boldsymbol{x_{ji1}}$ term $\boldsymbol{t_{i1}}$ in each document X_j will be changed by:

$$X_{ji}\hat{}=x_{ji1}+\sum_{i2=1,i2=i1}^{m}\boldsymbol{\delta_{i1,i2}}x_{ji2} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ \ldots\ \ldots\ldots \quad (5)$$

This step in fact updates the original term-based VSM by considering the semantic relatedness between each pair of terms and new text representation is called the ontology based VSM.

Example: table 1 and table 2 show simple example for modification of text representation with WordNet. Table 1 gives two documents in the traditional term based VSM. According WordNet, the terms *ball, football, basket ball* are semanticall y related to each other so we use equation(5) updating term weights for each document as shown in table 2

Doc	**Ball**	**football**	**basketball**	**food**
d1	**5**	**0**	**3**	**2**
d2	0	4	1	0

Table 1: A simple example for traditional term based VSM

Doc	**Ball**	**football**	**basketball**	**food**
d1	**7.4**	**6.4**	**7**	**2**
d2	4	4.8	4.2	0

Table 2: The representation table 1 data in ontology based VSM

Now we list the process how to calculate the term mutual information from the corpus.

The mutual information between two terms t_1 and t_2 can be calculated on the basis of ontology based VSM. Some techniques have been proposed by Mitra el at[6] and Cimino el at[10]. Here we adopted the ***cosine similarity*** to measure the term mutual information between their corresponding vectors.

$$Cos(t_1,t_2)=t_1.t_2/||t_1||.||t_2||\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots.(6).$$

$$Cos(t_1,t_2)=\sum_{j=1}^{n}x_{j1}x_{j2}/(\sum_{j=1}^{n}x_{j1}^{2})^{1/2}(\sum_{j=1}^{n}x_{j2}^{2})1/2\ldots\ldots\ldots\ldots(7)$$

Where x_{j1} and x_{j2} represents the term weights of t_1 and t_2 in the document $\boldsymbol{X_j}$ in the ontology based VSM.

According to the above cosine measure, the similarity of each pair of terms in the given corpus can be computed. The following table(Table 3) ten relative important similar terms with respect to our corpus, and the similarity obtained by equation (7)

4.3 TERM FREQUENCY AND WEIGHT IN DOCUMENT

Thus far, scoring has hinged on whether or not a query term is present in a zone within a document. We take the next logical step: a document or zone that mentions a query term more often has more to do with that query and therefore should receive a higher score. To motivate this, we recall the notion of a *free text query* introduced: a query in which the terms of the query are typed freeform into the search interface, without any connecting search operators (such as Boolean operators). This query style, which is extremely popular on the web, views the query as simply a set of words. A plausible scoring mechanism then is to compute a score that is the sum, over the query terms, of the match scores between each query term and the document. Towards this end, we assign to each term in a document a *weight* for that term, that depends on the number of occurrences of the term in the document. We would like to compute a score between a query term ***t*** and a document ***d***, based on the weight of ***t*** in ***d***. The simplest approach is to assign the weight to be equal to the number of occurrences of term ***t*** in document ***d***. This weighting scheme is referred to as *term frequency* and is denoted $\boldsymbol{tf_{t,d}}$, with the subscripts denoting the term and the document in order.

For a document ***d***, the set of weights determined by the ***tf*** weights above (or indeed any weighting function that maps the number of occurrences of ***t*** in ***d*** to a positive real value) may be viewed as a quantitative digest of that document. In this view of a document, known in the literature as the *bag of words model* , the exact ordering of the terms in a document is ignored but the number of occurrences of each term is material (in contrast to Boolean retrieval). We only retain information on the number of occurrences of each term. Thus, the document ``Mary is quicker than John'' is, in this view, identical to the document ``John is quicker than Mary''. Nevertheless, it seems intuitive that two documents with similar bag of words representations are similar in content. Before doing so we first study the question:

Terms	Term Similarity
(software ,hardware)	0.9240
(Arab, people)	0.9163
(baseball, sport)	0.8974
(space, science)	0.8948
(graphics, computer)	0.8365
(jaw, race)	0.7769
(orbit, satellite)	0.7514
(symmetric, circle)	0.7212
(team, players)	0.7028
(science, research)	0.6113

Table 3 Term mutual information calculated

All words in a document equally important? We looked at the idea of *stop words* - words that we decide not to index at all, and therefore do not contribute in any way to retrieval and scoring.

V. IMPLEMENTATION DETAILS

To test our algorithm we have coded it and the older one in Java Programming language. The rest of this section describes about the input dataset and cluster quality metric entropy which we have used in our paper.

A. Input Dataset

For testing purpose we have used both a synthetic dataset and a real dataset.

1) Synthetic Dataset: This dataset contains a total 12 classes from different books and articles related to different fields such as art, philosophy, religion, politics etc. The description is as follows.

B. Entropy

Entropy measure uses the class label of a document assigned to a cluster for determining the cluster quality. Entropy gives us the information about the distribution of documents from various classes within each cluster. An ideal clustering solution is the one in which all the documents of a cluster belong to a single class. In this case the entropy will be zero. Thus, the smaller value of entropy denotes a better clustering solution.

Given a particular cluster Sr of size Nr, the entropy [1] of this cluster is defined to be

$$(\quad)=-1/\quad \sum^{q}_{i=1} (\quad / \quad)/ \quad ?(\quad / \quad) \ldots\ldots\ldots\ldots (8)$$

where q is the number of classes available in the dataset, and is the number of documents belonging to the ^{h} class that were assigned to the ? cluster. The total entropy will be given by the following equation

$$= \sum_{i=1,}^{k} (\quad / \quad) \ (\quad)=1 \ldots\ldots\ldots\ldots (9)$$

VI. EVALUATION OF THE CRITERION FUNCTION FOR AGGLOMERATIVE CLUSTERING

Our first set of experiments was focused on evaluating the quality of the clustering solutions produced by the internal criterion function when they were used to guide the agglomeration process. The entropy results for the various datasets and criterion function for 10- and 20-way clustering solutions are shown in Table 4. The results in this table are provided primarily for completeness and in order to evaluate the various criterion functions we actually summarized these results by looking at the average performance of each criterion function over the entire set of datasets.

6.1 Evaluation of k-way clustering via repeated bisections

Our second set of experiments was focused on evaluating the clustering solutions produced by the various criterion functions when the overall solution was obtained via a sequence of cluster bisections (RB). In this approach, a *k*-way solution is obtained by first bisecting the entire collection. Then, one of the two clusters is selected and it is further bisected, leading to a total of three clusters. This step of selecting and bisecting a cluster is performed $k – 1$ times leading to the desired k-way clustering solution. Each of these bisections is performed so that the resulting bisection optimizes a particular criterion function. However, the overall k-way clustering solution will not necessarily be at a local optimum with respect to that criterion function. The key step in this algorithm is the method used to select which cluster to bisect next.From the above tables and figures gives evidence that, if the number of partitions increases The quality of clustering solution also increases. We are experiments k=5,k=10,k=15,k=20 Ways of clustering and corresponding entropy values. In our method the entropy values decreases when number of clusters increases that will gives the quality of the clustering. The performance of our algorithm improves when the number of clusters increases

	10-way clustering	**20-way clustering**
Name	**Internal criterion function(I2)**	**Internal criterion function(I2)**
Tr31	0.270	0.196
Tr41	0.262	0.187
Re0	0.455	0.405
Re1	0.470	0.383
ka1	0.472	0.418
k1a	0.180	0.163
wao	0.448	0.397
fibs	0.458	0.396
hitech	0.726	0.673
la1	0.580	0.530
la2	0.540	0.515
reviw	0.442	0.374

TABLE 4: Entropy values for various datasets criterion function for the clustering solution obtained via agglomerative clustering

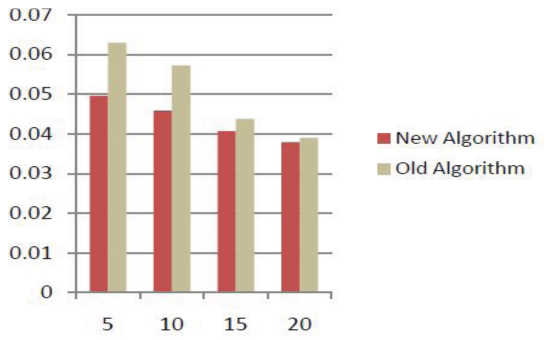

Fig 1: Variations of entropy Vs and number of clusters for data sets re0(no. of classes 15)

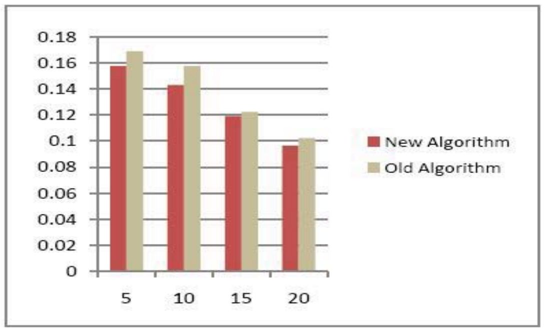

Fig 2: Variations of entropy Vs and number of clusters for data set re1(no. of classes

VII. RESULTS

In this paper we have successfully proposed and tested a new algorithm that can be used for accurate document clustering. We know that the most of the previous algorithms have a relatively greater probability to trap in local optimal solution. Unlike these the proposed algorithm has a very little chance to trap in local optimal solution, and hence it converges to a global optimal solution. In this algorithm, we have used a completely new analytical approach for initial clustering which refines result and it gets even more refined after the completion of refinement process. The performance of the algorithm enhances with the increase in the number of clusters. The clustering tool CLUTO is used to cluster low and high dimensional data, which improves the performance of our criterion function for different types of datasets with increasing number of clusters and also improves quality of the cluster.

REFERENCES

[1] Y. Zhao and G. Karypis, "Criterion functions for document clustering: Experiments and analysis," Technical Report #01-40, University of Minnesota, 2001.

[2] Cui, X.; Potok, T.E.; Palathingal, P., "Document clustering using particle swarm optimization," Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE , vol., no., pp. 185-191, 8-10 June 2005.

[3] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," IEEE Trans. Pattern Anal. Mach. Intell., vol. 24, no. 7, pp. 881-892, July 2002.

[4] M. Mahdavi and H. Abolhassani, "Harmony k -means algorithm for document clustering," Data Mining and Knowledge Discovery 2009.

[5] A.K. Jain and R. C. Dubes, " Algorithms for Clustering Data," Prentice Hall, 1988.

[6] S. Guha, R. Rastogi, and K. Shim, "Rock: A robust clustering algorithm for categorical attributes," Information Systems, vol. 25, no. 5, pp. 345-366, 2000.

[7] S. Guha, R. Rastogi, and K. Shim, "Cure: an efficient clustering algorithm for large databases," SIGMOD Rec., vol. 27, no. 2, pp. 73-84, 1998.

[8] G. Karypis, Eui, and V. K. News, "Chameleon: Hierarchical clustering using dynamic modeling," Computer, vol. 32, no. 8, pp. 68-75, 1999

[9] E. H. Han, G. Karypis, V. Kumar, and B. Mobasher, "Hypergraph based clustering in high-dimensional data sets: A summary of results," Data Engineering Bulletin, vol. 21, no. 1, pp. 15-22, 1998.

[10] B. Larsen and C. Aone, "Fast and effective text mining using linear-time document clustering," Knowledge Discovery and Data Mining, 1999, pp. 16-22.

[11] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of document clustering techniques," KDD Workshop on Text Mining Technical report of University of Minnesota, 2000.

[12] Y. Zhao and G. Karypis, "Empirical and theoretical comparisons of selected criterion functions for document clustering," Mach. Learn., vol. 55, no. 3, pp. 311-331, June 2004.

[13] Y. Zhao and G. Karypis, "Evaluation of hierarchical clustering algorithms for document datasets," in CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management. ACM Press, 2002, pp. 515-524.

[14] G. Salton, "Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer," Addison-Wesley, 1989.

[15] Y. Zhao, G. Karypis, and U. Fayyad, "Hierarchical clustering algorithms for document datasets," Data Mining and Knowledge Discovery, vol. 10, no. 2, pp. 141-168, March 2005.

[16] http://glaros.dtc.umn.edu/gkhome/fetch/sw/cluto/datasets.tar.gz

[17] C. Fellbaum wardNet an electronical lexical data base MIT press.1998

[18] M.F Porter, An Algorithm For suffix stripping program 14(3)(1980),pp 22-31.

[19] M.Sabou learning web service ontologics automatic extraction method and its evaluation, ontology learning from text .methods Applications and evaluations IOS press,2005