International Journal of Research Studies in Computer Science and Engineering (IJRSCSE)
Volume 6, Issue 1, 2019, PP 9-18
ISSN 2349-4840 (Print) & ISSN 2349-4859 (Online)
DOI: http://dx.doi.org/10.20431/2349-4859.0601002
www.arcjournals.org

A Comparison-Based Soft Clustering Algorithm for Documents

Ganesh Yadav[1], Vipul Kumar Verma[2]

[1]*Assistant Professor, Department of CSE, IIMT Greater Noida, India.*

[2]*Assistant Professor, Department of CSE, IIMT Greater Noida, India.*

Corresponding Author:** ***Ganesh Yadav, *Assistant Professor, Department of CSE, IIMT Greater Noida, India.*

Abstract: *Data document clustering is an most important tool for searching document such as Web search engines. Clustering data documents enables the accessor to have a good overall view of the information contained in the documents that he has. However, existing clustering algorithms faces from various aspects; complex clustering algorithms (where each document belongs to exactly one cluster) cannot detect the multiple themes of a document, while flexible such as soft clustering algorithms (where each document can belong to multiple clusters) are usually inefficient. We propose CSCA (Comparison-based Soft Clustering), an efficient soft clustering algorithm based on a given similarity measure. CSCA requires only a similarity measure for clustering and uses randomization to help make the clustering efficient. Comparison with existing complex hard clustering algorithms like K-means and its variants shows that CSCA is both effective and efficient.*

Keywords: *K-Means, cluster centroids, stop words, UCI document*

1. INTRODUCTION

Clustering is an important tool in data mining and knowledge discovery. The ability to automatically group similar items together enables one to discover hidden similarity and key concepts, as well as summarize a large amount of data into a small number of groups. This enables the users to comprehend a large amount of data.

Some search engines have pre-defined subjects that are used to categorize the output (for instance, yahoo.com). However, few search engines provide a dynamic clustering mechanism – i.e. clustering algorithms that are applied only to the resulting documents of the query. We believe that this is an important service for any search engine over the Web and is highly beneficial to users. Thus, in this paper we explore dynamic clustering for documents.

A challenge in document clustering is that many documents contain multiple subjects. For instance, a Web page discussing the University of Memphis's research on wild tiger fits under the categories of both "wild animals" and "Memphis". Thus, the clustering algorithm should discover this and put the document under both clusters. This suggests the use of "soft clustering" algorithm – an algorithm that allows a document to appear in multiple clusters. This can help

users to discover multiple themes in a document – by looking at the multiple clusters that a document belongs to. Soft clustering can also help form clusters containing combination of existing topics. For instance, we might want to have a separate cluster of documents about the University's research on wild tigers. This is possible if documents can fall into multiple clusters.

Many soft clustering algorithms have been developed and most of them are based on the Expectation-Maximization (EM) algorithm [6]. They assume an underlying probability model with parameters that describe the probability that an object belongs to a certain cluster. Based on the data given, the algorithms try to find the best estimation of the parameters. However, a drawback of such algorithms is that they tend to be computationally expensive.

In this paper, we take a different approach. Instead of assuming an underlying probability model, we only assume that we are given a similarity function $f(x, y)$, which given documents x and y, returns a

value between 0 and 1 denoting the similarity of these two documents. We develop CSCA (Similarity-based Soft Clustering), a soft-clustering algorithm based on the similarity function given. CSCA is similar to many other soft clustering algorithms like fuzzy C-means [2]. That is, it starts out with a carefully selected set of initial clusters, and uses an iterative approach to improve the clusters. At the end, CSCA produces a set of clusters with each document belonging to several potential clusters. This approach only requires a similarity function to be defined properly, and does not rely on any underlying probability assumptions (other than those made by the similarity function). To speed up execution, we propose using randomization to speed up the algorithm. This allows CSCA to overcome problems of standard soft clustering algorithms mentioned above, without paying any price in efficiency (in fact, we outperform K-means based algorithm in many cases).

The rest of the paper is organized as follows: Section 2 summarizes related work in the field. Section 3 describes CSCA in more detail. Section 4 provides some preliminary experimental results. Section 5 outlines future direction of this work.

2. RELATED WORK

Clustering is important in many different fields such as data mining, image compression and information retrieval .It provides an extensive survey of various clustering techniques. In this section, we highlight the work most related to our research.

We can divide clustering algorithms into hard and soft clustering algorithms. According to [10], there are four different kinds of clustering algorithms: hierarchical, partition, model fitting and density based. These algorithms form clusters by putting each item into a single cluster. Soft clustering allows each item to associate with multiple clusters, by introducing a membership function W_{ij} between each cluster-item pair to measure the degree of association. Expectation-maximization [6] serves as the basis of many soft-clustering algorithms. A good survey of such algorithms can be found in [1].

Many clustering techniques have been used for document clustering. Most of the early work [7, 15] applied traditional clustering algorithms like K-means to the sets of documents to be clustered. Willett [21] provided a survey on applying hierarchical clustering algorithms into clustering documents.

Cutting et al. [4] proposed speeding up the partition-based clustering by using techniques that provide good initial clusters. Two techniques, Buckshot and Fractionation are mentioned. Buckshot selects a small sample of documents to pre-cluster using a standard clustering algorithm and assigns the rest of the documents to the clusters formed. Fractionation splits the N documents into *'m'* buckets where each bucket contains N/m documents. Fractionation takes an input parameter ρ, which indicates the reduction factor for each bucket. The standard clustering algorithm is applied so that if there are 'n' documents in each bucket, they are clustered into n/ρ clusters. Now each of these clusters is treated as if they were individual documents and the whole process is repeated until 'K' clusters are left.

Most of the algorithms above use a word-based approach to find the similarity between two documents. In [22] a phrase-based approach called STC (suffix-tree clustering) was proposed. STC uses a suffix-tree to form common phrases of documents enabling it to form clusters depending not only on individual words but also on the ordering of the words.

Various other clustering techniques have been applied to document clustering. This includes using association rules and hypergraph partitioning [12], self-organizing maps [16], neural networks [19, 14], and EM-based techniques [9].

3. DESCRIPTION OF CSCA

CSCA aims at providing soft clustering on a set of documents based on a given similarity measure. It has the following goals:

- Enable soft clustering: documents can be clustered into multiple clusters.
- Efficient: CSCA should be able to run faster than traditional hard clustering algorithms.
- Cluster discovery: the algorithm should be able to find clusters that hard clustering algorithms cannot find.

- Handle outliers: the algorithm should be robust against outliers.

CSCA requires a similarity measure between documents: that is, given two documents *x* and *y*, there is a function $0 \le f(x, y) \le 1$ which returns how similar *x* and *y* are. It also requires a number *k*, which denotes the number of clusters that the user is expecting. Note that CSCA can decide to produce a different number of clusters, depending on the input documents.

CSCA produces a set of clusters at the end. Each cluster *c* is denoted by a set of documents called *cluster centroids*. The centroids serve two purposes: to define the set of documents most representative of the cluster, and to determine the degree of membership between *c* and each document. A measure *m(c, x)* is defined to represent how similar a document *x* is to cluster *c*. Intuitively, the documents within each cluster centroid should be close to one another: that is, having *f(x, y)* large for *x, y* belonging to the same cluster centroid. We will discuss how "large" is quantified later.

CSCA can be broadly divided into four steps: a pre-processing step to clean up and transform the data; an initial cluster generation step to initialize clusters and remove outliers; an iterative step to build clusters; and a post-processing step to present the results. Each step is described below:

3.1. Pre-Processing

In this step each document is transformed into a structure that will be used by the similarity function *f()*. One such representation is a vector, with each dimension denoting the presence/absence of a certain word in that document. In addition, we remove all the *stop words* (like articles, propositions and auxiliaries verbs) that are not helpful in clustering.

3.2. Initial Cluster Generation

At this step the input is analyzed, initial clusters are produced and outliers are removed.

The first thing for CSCA to do is to decide what constitute as "similar" documents. Essentially, we need to find a threshold value λ such that two documents are considered similar if and only if $f(x, y) \ge \lambda$. Since CSCA is designed to adapt to different similarity measures *f*, it is not reasonable for the user to supply a value for λ. As a result, CSCA determines the appropriate value of λ based on the input documents. The value of λ can neither be too high, such that no documents will be clustered at the end; nor too low, such that all documents will be clustered into one cluster. Thus, the algorithm chooses λ such that half[1] of the documents are assigned at least to one cluster centroid. This is done by the following method:

- Pick a set of *k* documents, assigning each one as the initial cluster centroid of a cluster.
- Pick λ as the largest value such that for half of the documents *q* in the data set, there exists a *p* such that $f(p, q) \ge \lambda$, $p \in C, q \in D$ where *C* is the set of cluster centroids and *D* is the document set.

This can be done by calculating all the similarity values $f(p, q), \forall p \in C, \forall q \in D$ and sorting them.

One drawback of this approach is that outliers can easily be picked as centroids, rendering the clustering method ineffective. To remedy that, the algorithm makes use of the threshold value λ selected. After λ is picked, the chosen cluster centroids are re-examined to make sure that there is at least one document similar to it (i.e. with $f() \ge \lambda$). Otherwise, the document is viewed as an outlier and is discarded, and replaced with an alternate document chosen as a centroid (the same test is applied to the new document to ensure that it is not an outlier itself).

This ensures that at least half of the documents are close to at least one of the clusters, so that enough interesting clusters can be found. An issue here is how the initial cluster centroids are picked. The simple way is to pick a random set of documents. However, since the initial cluster centroids can have

a significant effect on the algorithm, it pays to be more careful. We want to avoid picking too many cluster centroids that are close to one another (so they should actually belong to the same cluster). One way to overcome it is to start with picking a random document as the first centroid, and then pick the document that is least similar to it as the second centroid. Subsequent centroids are chosen such that they are farthest away from those centroids that are already picked.

To make the algorithm more robust to the initial choice of cluster centroids, CSCA starts with *2k* instead of *k* initial clusters. This makes the algorithm more flexible to find the right number of clusters.

3.3. Recursive Step

In this step, clusters are refined. Since CSCA uses cluster centroids as representative of each cluster, this step examines each cluster and decides whether the centroids should change. The algorithm terminates when no more such changes are made.

To determine whether a document should be in a cluster centroid, we need a measure of similarity between the document and the cluster. Thus, we define a measure *m(c, x)* that denotes the similarity of document *x* for cluster *c*. It is defined as the average similarity of *x* for the documents in the current centroid of cluster *c*. At each iteration, the value of each *m(c, x)* is re-calculated. If the value is larger than the threshold λ, then document *x* is put into the centroid of *c*. However, if for any document $y \in c$, the new value of $m(c, y) \prec \lambda$, then it is removed from that cluster centroid

One addition step CSCA needs to perform during each iteration is the merging of clusters. Though we are careful in choosing initial clusters centroids, similar documents can still be chosen as different cluster centroids. Since documents can be in multiple clusters, the cluster centroids for those clusters will converge in subsequent iterations. Therefore, we need to merge clusters with similar cluster centroids. We experimented with various schemes and found that merging two clusters when at least half the documents in one cluster centroid appear on the other gives the best results. Two clusters are also merged if one is a subset of the other.

The drawback of the above is that during every iteration, the value of *m(c, y)* has to be recalculated for each *(c, y)* pair. This can very time consuming. However, we observed that in most cases membership for the cluster centroid changes only for documents that are already close to the threshold value. Thus, we can speed up the algorithm by using randomization. Rather than calculating every *m(c, y)* pair, CSCA recalculates the new value of *m(c, y)* with the probability of $m(c, y)/\lambda$; i.e. the chance of recalculating the similarity measure is proportional to how close it is to the threshold. This cuts down on many unnecessary calculations while maintaining the cluster quality.

3.4. Displaying Clusters and Keywords

We need to display the final clusters at the end of the algorithm. Each cluster *c* is represented by the cluster centroids as the representatives. Moreover, for each document *y*, *m(c, y)* is used as the measure of similarity. Thus for each cluster, the documents can be sorted by this value to determine its association with the cluster and the results can be displayed accordingly.

One final step is to annotate each cluster with keywords (terms) so that summary information about the clusters can be provided. For each cluster *c*, we keep track of the terms that appear in the documents of that cluster's centroid. For each term in the documents in the centroid, we calculate two values:

- *n*, the number of documents in the cluster centroid that it appears in.
- $w = \sum m(c, y) \ , \forall y \in X$, where *X* is the set of all the documents in which the word appears.

We ordered the terms by *n * w*, and displayed the top 6-7 of them. We experimented with different formulae and found this way of calculating keywords for a cluster as the best.

4. EXPERIMENTS & RESULTS

This section describes the results of the various experiments with CSCA. In order to evaluate the performance, we compared CSCA with other algorithms like K-Means, Fractionation and Buckshot [4].

The adhoc test consists of 4000 documents downloaded from the Web. We downloaded 2000 documents from various categories like Food, Cricket, Astronomy, Clustering, Genetic Algorithms, Baseball, Movies, Virus, XML, Albert Einstein, Salsa (Dance), Salsa (food), Health care, stocks, Cancer, Tigers and the Olympics. We downloaded another 2000 documents from the UCI KDD archive [18], which has documents from 20 different news groups.

In our experiments we used "Tanimoto coefficient" [5, 20] as the similarity measure. It is defined as follows: If $n1$ is the number of terms in the first document and $n2$ is the number of terms in the second document and m is the number of common terms then the similarity measure between the two documents is given by $\frac{m}{n1+n2-m}$. Note that CSCA does not preclude the use of other measures. We chose the Tanimoto coefficient because of its simplicity.

All the experiments described in this section are carried out on a 333 MHz, 196 MB RAM PC. We did experiments on different document sets of different sizes. We ran the algorithm to get the clusters and tested the effectiveness of clustering (the types of clusters formed), both qualitatively and quantitatively. We also compared the execution times of all the algorithms for document sets of different sizes.

4.1. Effectiveness of Clustering

We did many experiments with the document sets of different sizes that are taken from the above-mentioned test bed. All the algorithms were run to produce the same number of clusters with same input parameter settings. CSCA formed clusters for each of the different categories in the document sets, while the other algorithms (K-Means, Fractionation and Buckshot) did not. In addition, the other algorithms formed clusters with documents of many categories that are not related to each other.

To test the effectiveness of our approach, we deliberately downloaded some documents that are related to more than one of the categories listed above (For example, documents about baseball movies which include both baseball and movies). K-Means, Fractionation and Buckshot formed only clusters about baseball and Movies and did not form a cluster for documents related to both categories. However, CSCA formed a cluster for baseball-movies and put the documents related to baseball-movies in that cluster. This shows the effectiveness of our method compared to other algorithms.

Figure 1 shows sample output from one of the experiments with a document set of 500 documents picked from 12 of the categories mentioned above. The categories include Food, agents, XML, Jordan (the middle east country), Genetic algorithms, baseball, movies, Astronomy, Michael Jordan, Consciousness, Virus and baseball-movies.

All the algorithms were run with an input number of 12 clusters. Other algorithms formed 12 clusters but they did not form a cluster for baseball movies. CSCA formed a cluster for that as shown in the above table. Since CSCA starts with twice the actual number of clusters and combines the clusters, we also tested other algorithms with twice the number of clusters as input.

However, they failed to produce the correct clusters even with twice the actual number of clusters.

Cluster results: Common words and sample docs	Category
Movie, Baseball, camera, Costner, Crash, Days 1. Love of the Game 2. Kevin Costner	Baseball movies
Mutation, chromosomes, crossover, genetic algorithm, offspring 1. The genetic algorithm archive 2. Evolutionary computation group	Genetic Algorithms
Information, SGML, Xlink, Xpointer, group 1. XML: Linking Language 2. XML Xpointer requirements	XML
Hoax, area, email, address 1. Virus-hoax information	Virus

2. The great Virus-Hoax	
Food, cooking, course, Gourmet, enjoy 1. Cookbooks 2. Glen's Recipes links page	Food
Philosophers, animals, comparative, conscious, mind, animal 1. Online papers on consciousness 2. Animal Consciousness	Consciousness
Line, agents, autonomous, central, communication 1. Agent or Program 2. Software Agents: An Overview	Agents
Awards, director, English, love, Oscar, Patient, Years 1. The Oscar's page 2. The 69th Academy awards	Movies
Galaxy, Milky, Review, Solar, Stars, Sun, System 1. Introduction to Astronomy 2. NASA Watch: Space news	Astronomy
Michael, Bulls, career, Chicago, Finals, game, greatest 1. History of Bulls 2. Michael Jordan's official web page	Michael Jordan
Leading, April, average, bases, ball, batsman, batting 1. MSU's baseball news 2. History of baseball	Baseball

Fig1. *Clusters formed by CSCA for a sample document set*

The table in the figure 2 compares the clusters formed by different algorithms

Category	CSCA	K-Means	Buckshot	Fractionation
Movies	Yes	Yes	No	Yes
Virus	Yes	Yes	No	No
Baseball	Yes	Yes	Yes	Yes
Baseball Movies	Yes	No	No	No
Food	Yes	No	No	Yes
astronomy	Yes	No	Yes	No
Jordan (country)	Yes	Yes	No	No
Michael Jordan	Yes	No	Yes	No
Agents	Yes	Yes	Yes	Yes
Consciousness	Yes	Yes	No	Yes
Genetic algorithms	Yes	Yes	Yes	Yes
ML	Yes	Yes	No	No

Fig2. *Comparison of clusters formed by different algorithms*

We also measured the effectiveness of the clustering algorithm quantitatively. We compared the clusters formed by the documents against the documents in the original categories and matched the clusters with the categories one-to-one. For each matching, we counted the number of documents that are common in corresponding clusters. The matching with the largest number of common documents is used to measure the effectiveness. This matching can be found by a maximum weight bipartite matching algorithm [17]. We return the number of documents in the matching. The more documents that are matched, the more they resemble the clusters are to the original categories. For our algorithm, and for the purpose of this comparison, we assigned each document to the cluster that has the largest similarity value.

Figure 3 shows the number of matches with the original categories for different algorithms averaged on 10 different sets of documents. We can clearly see that CSCA outperforms the other algorithms in effectiveness.

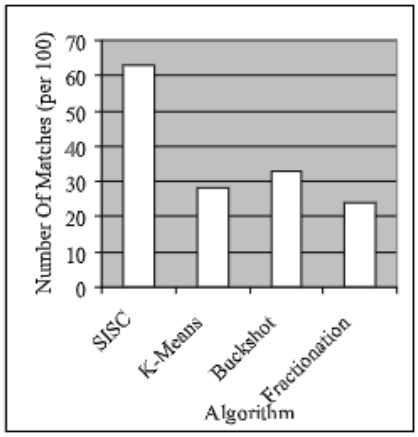

Fig3. *Comparison of quality of clusters by different clustering algorithms*

Next we turn to the data set downloaded from the UCI archive, which contains data from various newsgroups. The newsgroups contain topics like atheism, computer graphics, Mac hardware, pc hardware, x-windows, basketball, hockey, cryptography, electronics, space and Christianity. Due to limitation of space, we only show the result of cluster quality comparison in figure 4.

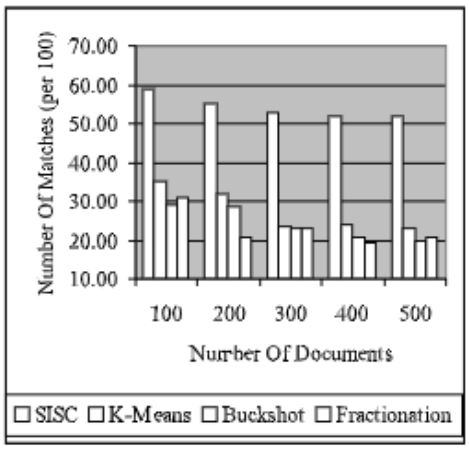

Fig4. *Comparison of quality of clusters by different algorithms on UCI document set*

The figure shows that our algorithm worked well with the data from the UCI document archive and clearly outperformed the others.

4.2. Execution Time

We also measured the execution time of various algorithms. Figure 5 gives the comparison of execution times.

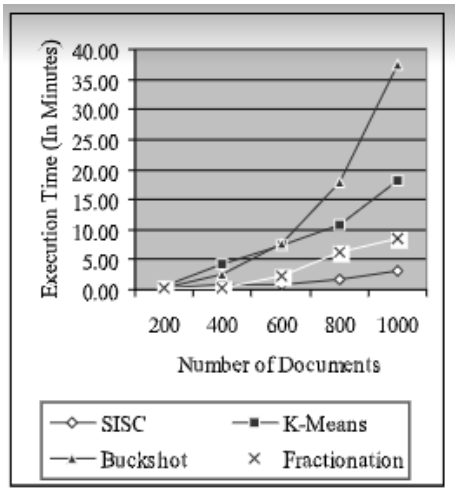

Fig5. *Execution times of various clustering algorithms*

As the graph shows, CSCA outperforms almost all other algorithms in execution time, especially as the number of documents increase.

To see how effective randomization is, we compared the execution times of our algorithm with and without randomization.

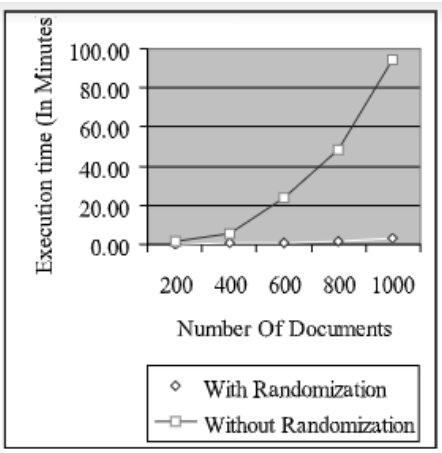

Fig6. *Execution times of CSCA with and without randomization*

We can observe from Figure 6 that introducing randomization cuts down the running time significantly. This shows the effectiveness of using the randomization approach.

Of course, we need to justify randomization by comparing the cluster quality. We use the quantitative measure to compare the two algorithms to see if they give similar results. Figure 7 shows that both algorithms give roughly the same number of matches. In fact, the difference is less than 5%; thus, we can justify the use of randomization to speed up the algorithm.

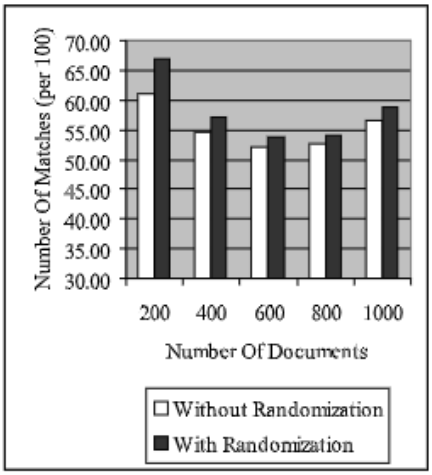

Fig7. *Comparison of cluster quality: randomization vs. no randomization*

4.3. Web Search Results

We also tested CSCA on the results from Web search engines. We downloaded documents returned from the Google search engine (www.google.com). After that, we clustered the documents using our algorithm and displayed the results. Limitation on space prohibits us from showing the full results. Here we show some clusters that we found by clustering the top 100 URLs returned from searching the term "cardinal". The categories correspond to the common usage of the word "cardinal" in documents over the Web (name of baseball teams, name of football teams, nickname for schools, a kind of bird, and a Catholic clergy). Figure 8 shows some of the clusters formed by CSCA.

Cluster results: Keywords and sample documents	Related topic
Priests, Cardinals, Catholic, Church, Deacons, Holy, Cardinal 1. Cardinals in the Catholic Church | Catholicpages.com 2. The Cardinals of the Holy Roman Church	Roman Catholic Church Cardinals library
NFL, Bay, beat, big, Boston, Bowl, Burke	Arizona Cardinals (American football

1. NFL Cardinals Forum 2. Arizona Cardinals	team)
Dark, Bird, Cardinal, Cardinals, colored, beak, female 1. Cardinal Page 2. First Internet Bird Club	Cardinal (bird)
Homer, Cardinals, Darryl, Drew, field, half, Ball 1. St.Louis Cardinals Notes 2. National League Baseball - Brewers vs. Cardinals	St.Louis Cardinals (American Baseball team)
Faith, politics, President, Cardinals 1. Who are the cardinals? 2. The Failure of Catholic Political Leadership	Catholic and politics

Fig8. *Clusters formed for search term "cardinals" by CSCA*

5. CONCLUSIONS

In this paper, we introduced CSCA; a soft clustering algorithm based on similarity measures, and applied it to document clustering. The algorithm introduces various techniques such as randomization to help make soft clustering efficient. Our experiments show that CSCA algorithm is able to discover clusters that cannot be detected by non-fuzzy algorithms, while maintaining a high degree of efficiency. We mentioned using clustering for Web search engines. Currently CSCA is practical to cluster about 600-800 documents, while maintaining reasonable (less than a minute) response time. We are looking at various techniques to improve this. We also believe that the randomization techniques can be used to improve fuzzy clustering in general. We would like to explore more possibilities along this avenue.

REFERENCES

[1] Alberto Munoz, Compound key word generation from document databases using a Hierarchical clustering ART Model, Intelligent Data Analysis, 1(1), Jan 1997. http://www-east.elsevier.com/ida/browse/96-5/ida96-5.htm

[2] J.L. Bezdek, Pattern Recognition With Fuzzy Objective Function Algorithms, Plenum Press, Nyew York, NY. 1981.

[3] M.S. Chen, J. Han, and P.S. Yu, Data Mining: An Overview from a Database Perspective, IEEE Transactions on Knowledge and Data Engineering, 8(6): 866-883, 1996.

[4] Dmitri Roussinov, Kristine Tolle, Marshall Ramsey and Hsinchun Chen, "Interactive Internet search through Automatic clustering: an empirical study", In Proceedings of the International ACM SIGIR Conference, pages 289-290, 1999.

[5] Dean, P. M. Ed., Molecular Similarity in Drug Design, Blackie Academic & Professional, 1995, pp 111 – 137.

[6] A.P. Dempster, N.M. Laird, and D. B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, Journal of the Royal Statistical Society, Series B, 39(1), 1-38, 1977.

[7] D. R. Hill, A vector clustering technique, in: Samuelson (Ed.), Mechanized Information Storage, Retrieval and Dissemination, North-Holland, Amsterdam, 1968.

[8] A.K. Jain, M.N. Murty and P.J. Flynn, Data Clustering: A Review, ACM Computing Surveys. 31(3): 264-323, Sept 1999.

[9] W.J. Krzanowski and F.H. Marriott, Multivariate Analysis: Classification, Covariance Structures and Repeated Measurements. Arnold, London, 1998.

[10] Kamal Nigam, Andrew Kachites Mccallum, Sebastian Thrun and Tom Mitchell, Text Classification from Labeled and Unlabeled Documents using EM. Machine Learning 39(2-3):103-134, 2000.

[11] Y. Linde, A. Buzo and R.M. Gray, An Algorithm for Vector Quantization Design, IEEE Transactions on Communications, 28(1), 1980.

[12] Jerome Moore, Eui-Hong (Sam) Han, Daniel Boley, Maria Gini, Robert Gross, Kyle Hastings, George Karypis, Vipin Kumar, and Bamshad Mobasher, Web Page Categorization and Feature Selection Using Association Rule and Principal Component Clustering, In Proceedings of seventh Workshop on Information Technologies and Systems (WITS'97), December 1997.

[13] M.N. Murty and A. K. Jain, Knowledge-based clustering scheme for collection management and retrieval of library books, Pattern recognition 28, 946-964, 1995.

[14] F. Murtagh, A survey of recent advances in hierarchical clustering algorithms, The Computer Journal, 26(4): 354--359, 1983.

[15] Baraldi, P. Blonda, A survey of fuzzy clustering algorithms for pattern recognition, Technical Report TR-98-038, International Computer Science Institute, Berkeley, CA, Oct 1998.

[16] J. J. Rocchio, Document retrieval systems – optimization and evaluation, Ph.D. Thesis, Harvard University, 1966.

[17] Douglass R. Cutting, David R. Karger, Jan O. Pedersen, John W. Tukey, Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections, In Proceedings of the Fifteenth Annual International ACM SIGIR Conference, pp 318-329, June 1992.

Citation: *Ganesh, Y& Vipul Kumar Verma (2019). A Comparison-Based Soft Clustering Algorithm for Documents. International Journal of Research Studies in Computer Science and Engineering (IJRSCSE), 6(1), pp.9-18. http://dx.doi. org/10.20431/2349-4859.0601002*