# Distributed Document Clustering Using PSO and Rough-KMeans Algorithm

Sereena.M.V, M Sharmila Kumari
MTech. Student, Head Of Department
Department of Computer Science and Engineering,
P. A. College of Engineering, Mangalore, 574153, India

***Abstract:*** In Data Mining, Clustering is one of the recently and challenging tasks it support wide range of unsupervised classification application. Fast and high-quality document clustering algorithms play an important role in helping users to effectively navigate, summarize and organize the information. In this paper, we present a hybrid Particle Swarm Optimization (PSO) + Rough K-means algorithm that performs fast document clustering. This proposed method utilizes the Hadoop and MapReduce framework which provides the distributed storage and analysis to support wide range of distributed applications. Here we use the BBC dataset for clustering which contains 1000 documents.

***IndexTerms* - Hadoop, MapReduce, PSO, Rough-KMeans, Document clustering.**

## I. INTRODUCTION

Clustering [1] or unsupervised learning is one of the most important fields of machine learning which splits the data into groups of similar objects helping in extraction or summarization of new information. It is used in a variety of fields such as pattern recognition and data mining. This research focuses on the application of clustering on data mining. Clustering is one of the major areas in data mining that has a vital role in answering to the challenges of every IT industry. Nowadays these document datasets are increasing tremendously which often referred to as Big Data. The problems of analysis on these datasets are referred to as a curse of dimensionality since they are often highly dimensional. Distributed computing plays a major role in data mining due to these reasons. Distributed Data Mining (DDM) has evolved as a hot research area to solve these issues. Document clustering has been investigated for use in a number of different areas of text mining and information retrieval. It is regarded as a major technology for intelligent unsupervised categorization of content in text form of any kind; e.g. news articles, web pages, learning objects, electronic books, even textual metadata. Document clustering groups similar documents to form a coherent cluster while documents that are different are separated into different clusters. The quality of document clustering in both centralized and decentralized environments can be improved by using an advanced clustering framework. Most of the conventional document clustering methods is designed for central execution which maintains a single large repository of documents where clustering analysis is performed. They require clustering to be performed on a dedicated node, and are not suitable for deployment over large scale distributed networks. These methods are based on the assumption that the data is memory resident, which makes them unable to cope with the increasing complexity of distributed algorithms. So there is an eminent need for mining knowledge from distributed resources. Therefore, specialized algorithms for distributed clustering have to be developed. Distributed clustering algorithms are needed to overcome the challenges in distributed document clustering.

In order to support data intensive distributed applications, an open source implementation based on Hadoop is used for processing of large datasets. MapReduce is a functional programming model for distributed processing over several machines. The important idea behind MapReduce framework is to map the datasets into a group of <key, value> pairs, and then reduce all pairs with the same key. A map function is performed by each machine which takes a part of the input and maps it to <key, value> pairs. This is then send to a machine which applies the reduce function. The reduce function combines the result for further processing. The outputs from the reduce function are then fed into the appropriate map function to begin the next round of processing.

In this proposed work, PSO, Rough-KMeans (PRKMeans) distributed document clustering method is formulated for better speedup and accuracy of document clustering. Along with these conceptual and algorithmic changes there is also a need to use any of the emerging frameworks for distributed analysis and storage. Today's enterprises use Big Data infrastructure for analytical engineered applications. One of the evolving technologies is the MapReduce methodology and its open source implementation is Hadoop [2]. This proposed method of document clustering is based on MapReduce methodology which improves the performance of document clustering and enables handling of large document dataset.

Section 2 highlights related work in the area of MapReduce based distributed document clustering using PSO and Rough-KMeans algorithm. Section 3 describes the proposed method. Result and analysis are done at Section 4. The paper concludes at Section 5.

## II. LITERATURE REVIEW

"Distributed Document Clustering Analysis Based on a Hybrid Method" [1] by J.E. Judith, J. Jayakumari propose a method called MR-PSO (mapreduce pso) and MR-KMeans.That is, author define the implementation of PSO and K-Means algorithms using

MapReduce Methodology.In paper “Design and Implement of Distributed Document Clustering Based on MapReduce”[2], published by Jian Wan, Wenming Yu1, and Xianghua Xu describes how the documents are clustered efficiently by using MapReduce concepts.Hadoop framework is used which provides better execution of distributed storage of documents. Author defins calculation of tf-idf and K-Means algorithm on MapReduce.“Hadoop: A New Approach for Document Clustering” [4] by Y.K. Patil and Prof. V.S. Nandedkar describe the design and implementation of Tf-Idf, K-means and Hierarchical clustering algorithms on Hadoop. In this, the authors first perform preprocessing and text processing of collected documents that are needed to find the Tf-Idf value.Then explain the K-means and Hierarchical clustering algorithm steps. The implementation code was written in Java language.“Parallel PSO Using MapReduce” by Andrew W. McNabb, Christopher K. Monson, and Kevin D. Seppi proposes a method called MapReduce PSO (MRPSO).It explains the particle swarm optimization algorithm which uses MapReduce concepts for parallel execution. “A Survey on K-mean Clustering and Particle Swarm Optimization” by Pritesh Vora, Bhavesh Oza”, present the K-Means algorithm and Particle Swarm Optimization Algorithm. The author concluding K-mean clustering is widely used to minimize squared distance between features values of two points reside in the same cluster and Particle Swarm Optimization Algorithm finds the optimum solution which gives better clustering efficiency.“Document Clustering Analysis Based on Hybrid PSO+K-means Algorithm” by Xiaohui Cui and Thomas E. Potok propose the method hybrid PSO+K-means algorithm which performs fast document clustering and also discussing comparison of PSO+K-means, PSO, K-means and other two hybrid clustering algorithms on four different text document datasets. “Some Refinements of Rough k-Means Clustering” by Georg Peters describes Lingras rough k-means clustering algorithm. And how this refined algorithm is applied to synthetic, forest and microarray gene expression data.” Comparative Study of K-Means, Pam and Rough K-Means Algorithms Using Cancer Datasets” by Parvesh Kumar and Siri Krishan Wasan explain the concepts of K-Means, PAM (Partitioning Around Medoids) and Rough K-Means Algorithms and all these are compared using cancer datasets.

## III. PROPOSED METHOD

This paper performs document clustering using PSO and Rough-KMeans algorithm. Here we use the BBC dataset which contains 5 categories like entertainment, sports, business, technology and politics data. The different step performed by this methodology is:

- Choosing a BBC document dataset to perform analysis. A variety of document datasets are publicly available and used for text mining research.
- Preprocess the document to reduce the number of attributes. The input text documents are transformed in to a set of terms that can be represented by a vector model.
- These terms are converted to document vector model by using tfidf value.
- These document vectors are clustered using a PSO and Rough-KMeans clustering algorithm based on MapReduce methodology (MR-PRKMeans).

The overview of the propose method is shown in **Figure.1**.In order to represent the documents the document datasets are preprocessed using common procedures like stemming using default porters [3] algorithm, stopword removal, pruning, removal of punctuations, whitespaces, numbers and conversion to lowercase letters. Each of the extracted terms represents a dimension of the feature space. This enhanced model as proposed includes two modules MR-PSO module, MR-KMeans module. Multiple MapReduce jobs are chained together to improve the quality and accuracy of clusters with reduction in execution time.

.

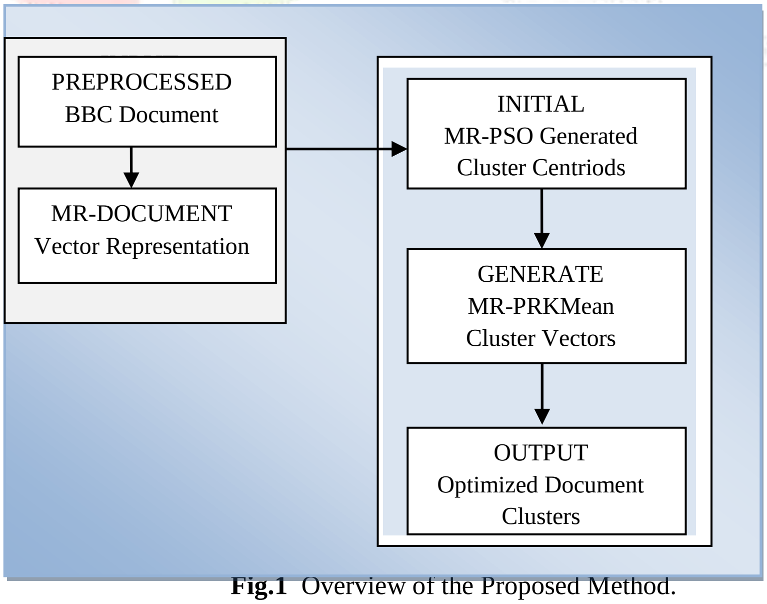


**Fig.1** Overview of the Proposed Method.

### 3.1 Representation of Document Vector Model using MapReduce

The pre-processed documents are represented as a vector using vector space model. Documents have to be transformed from full text version to a document vector which describes the content of the document. Let D= {d1, d2…..dn} be a set of documents and T = {t1, t2,....tm} be the set of distinct terms occurring in D. Each document is represented on a Document-Term Matrix (DTM) or document vector model. Each entry in the document-term matrix is the term weight. Terms that appear frequently in a small number of documents but rarely in other documents, tend to be more relevant and specific for that group of documents. These terms are useful for finding similar documents. To determine term weight, we use tf*idf weighting scheme [2], [4].

- Tf(t)=(Number .of times term t appear in a document) / (Total number of terms in the document).
- Idf(t)=log $_e$(Total no. of documents / Number of documents with term t)

Tfidf equation is given below.

$$tf * idf(d,t) = tf(d,t) * \log\left(\frac{|D|}{df(t)}\right) \qquad (3.1)$$

Where df(t) is the number of documents in which term t appears.

### 3.1.1 Map and Reduce function

The input to the Map function includes the document dataset. The document dataset is split and <key, value> pairs are generated for each individual text documents in the dataset. The key is the doc ID and the value is the terms in the document <DocID, term>. The Map function determines the term frequency of each term in the document. And writes DocID is the Key and stemming term and frequency are value to the reduce function. The list of terms and term frequency of each term in the document collection are given as input to reduce function and it finds the Idf value of each of the document and finally determines the Tfidf value using the above equation to form a Document vector model. The algorithm for Document vector model based on MapReduce is shown in **Figure. 2**.

```
Function Map (key: DocID, val: term)
For each docID in documents
         //Extract the term frequency of each term in the document
         Extract (term, docID, termcount)
End for
        new_key = (DocID)
        new_val = term,termcount
        emit (new_key, new_val)
end function
Function Reduce (key:DocID,valList:termcount)
for each term in documents
         //Extract a list of tfidf for each term from document   collection
         Extract (DocID, list (term, termcount)
end for
        new_key = DocID
         new_val = (tfidf) value
         emit (new_key, new_val)
end function
```

**Fig.2** Pseudo code for document vector representation based on MapReduce.

### 3.2 MapReduce Particle Swarm Optimization(MR-PSO) module

PSO is an iterative global search method. In the PSO document clustering algorithm, the multi-dimensional document vector space is modeled as a problem space. Each term in the document dataset represents one dimension of the problem space. Each document vector can be represented as a dot in the problem space. The whole document dataset can be represented as a multiple dimension space with a large number of dots in the space. One particle in the swarm represents one possible solution for clustering the document collection [5]. Therefore, a swarm represents a number of candidate clustering solutions for the document collection. Each particle maintains a matrix Xi = (C1, C2… Ci... Ck), where Ci represents the ith cluster centroid vector and k is the number of clusters. According to its own experience and those of its neighbors, the particle adjusts the centroid vector' position in the vector space at each generation. The average distance of documents to the cluster centroid is used as the fitness value to evaluate the solution represented by each particle. The fitness value [6] is measured by the equation below:

$$f = \frac{\sum_{i=1}^{Nc}\left(\frac{\sum_{j=1}^{Pi} dis(Oi, dij)}{Pi}\right)}{Nc} \qquad (3.2)$$

where dij denotes the jth document vector, which belongs to cluster i; Oi is the centroid vector of the ith cluster; dis(Oi, dij) is the distance between document dij and the cluster centroid Oi.; Pi stands for the number of documents, which belongs to cluster Ci; and Nc stands for the number of clusters. Here NC=5.The velocity and position of new particle are updated based on the following equations,

$$V=w*V+c1*rand1*(PB-Ci)+c2*rand2*(GB-xid) \qquad (3.3)$$

$$xid=xid+vid \qquad (3.4)$$

Where vid is the velocity and xid is the position of the particle w represents the inertia weight PB is the personal best position where the particle experience the best fitness value and GB is the global best position where the particle experience a global fitness value c1 and c2 denotes the acceleration coefficient, rand1 and rand2 are random values with the values between (0, 1).The personal, global best positions and velocity of each particle are updated based on Equations. The two major operations are performed in this module; such as fitness evaluation and particle centroids updates. PSO module for generating optimal centroids for clustering is summarized as follows,

*(1) At the initial stage, each particle randomly chooses k different document vectors from the document collection as the initial cluster centroid vectors.*

*(2) For each particle:*

*(a) Assign each document vector in the document set to the closest centroid vector.*

*(b) Calculate the fitness value based on equation.*

*(c) If the fitness value is better than the previous personal best fitness value (PB) set current value as the new PB.*

*(1) Update the particle position and velocity according equations.*

*(3) Repeat step (2) until maximum number of iterations is exceeded.*

*(4)Choose the particle with the best fitness value of all the particles as the global best (GB) fitness value.*

*And GB value gives the optimal cluster centroids.*

### 3.2.1 Map and Reduce function

The input to the Map function includes the tf-idf representation of documents stored in local file. Map function splits the documents and <key, value> pairs are generated for each individual text document in the dataset. In the first module the MapReduce job is used for generating optimal centroids using PSO [7], [8]. The map function evaluates the fitness of each particle in the swarm. All the information about the particle such as particleID, Cluster vector(C), Velocity vector(V), Personal Best Value(PB), Global Best Value(GB) are determined. The particleID represents the key and the corresponding content as the value the particle swarm is retrieved from the distributed storage. For each particleID the map function extracts centroids vectors and calculates the average distance between centroids vector and document vector. It returns a new fitness value based on global best values to the reduce function. The reduce function aggregates the values with the same key and updates the particle position and velocity. The reduce function emits the global best centroids as the optimal centroids for the next module. The pseudo code [9] for the determination optimal centroids is summarized in **Figure.3**.

```
Function Map (key: particleID, value:particle)
    ExtractInfo($C_i$, V, PB, GB)
    read (swarm)
    for each $c_i$ in $C_i$
        for each particle in swarm
            $C_i$=extractCentroids(particle)
            PID=extract id(particle)
            minDist=return MinDistance(particle, $C_i$)
            Cid=centroid //minimum distance
            newKey=(PID, Cid)
            newValue=min_dist
            emit(newKey, newValue)
        end  for
end  function
Function Reduce(key:(PID, Cid, ValList:min_dist)
  for each $c_i$ in C
        for each value in ValList
            count=count+1
            sumDist=sumDist+minDist
        end for
            avgDist=sumDist/count
            p=avgDist
        if fitness(p) is better than fit(PB)then
            PB=p
            PB=GB
        $newV_i$= 0.5*$V_i$+($rand_1$*0.08)*(PB-Ci)+($rand_2$*0.07)*(GB-Ci)
        $newC_i$= $C_i$+$newV_i$
        update (particle, $newV_i$, $newC_i$)
  end for
        docID=PID
        CentroidID=newCi//optimal centroids
emit (key:docID, value:CentroidID)
end function
```

**Fig.3** The pseudo code for the determination optimal centroids (PSO).

### 3.3 MapReduce Rough-KMeans (MR-RKMeans) Module

For the first iteration the clustering process gets the optimal initial cluster centroids from PSO and for the other iteration it gets cluster centroids from the last MapReduce output. The clustering process works according to the MapReduce program for similarity calculation, assignment of document to clusters and recalculation of new cluster centroids.

Rough set [10] is a mathematical tool used to deal with uncertainty. We use rough sets; here, a cluster is represented by a rough set based on a lower approximation and an upper approximation. The rough K-means algorithm can be stated as follows:

1. Select initial clusters of n objects into k clusters.
2. Assign each object [11] to the Lower bound (L(x)) or upper bound (U(x)) of cluster/ clusters respectively as: For each object v, let d (v,xi) be the distance between itself and the centroid of cluster xi. The difference between dis(v,xi) - d(v,xj), 1≤ i, j ≤ k is used to determine the membership of v as follows:
   - If dis(v,xi) - dis(v,xj) ≤ thersold, then v∈U(xi) & v∈U(xj).The v will not be a part of any lower bound.
   - Otherwise, v∈L(xi),such that d(v,xi) is the minimum for 1≤ i ≤ k. In addition, v∈U(xi).

3. For each cluster xi re-compute center [12] according to the following equations the weighted combination of the data points in its lower_bound and upper_bound.

$$X= \left\{ \left(wlower * \frac{\sum_{v\in L(x)} Vj}{|\mathbf{L(x)}|}\right) + \left(\text{Wupper} * \left(\frac{\sum_{v\in U(x)-L(x)} Vj}{|\mathbf{U(x)-L(x)}|}\right)\right) \qquad \textbf{if } |\boldsymbol{U(x)} - \boldsymbol{L(x)} \neq \emptyset| \right.$$

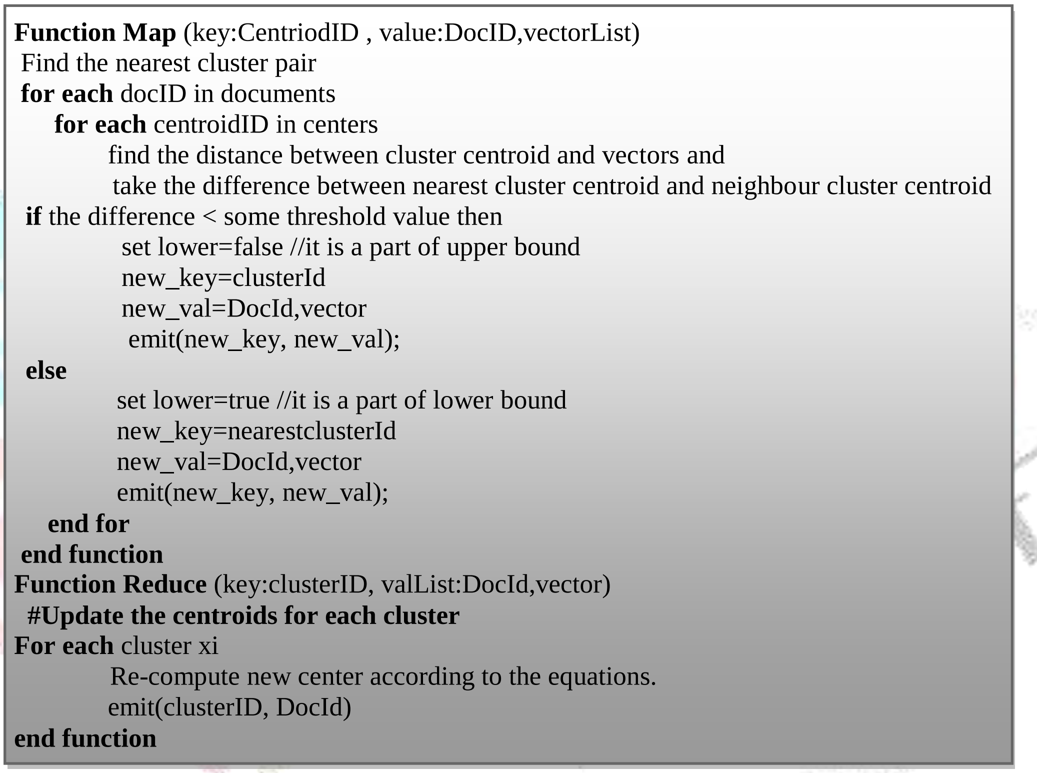
$$\begin{cases}(Wlower * (\sum_{v\in L(x)} Vj)) & otherwise\end{cases} \quad (3.4$$

Where 1≤ j ≤ k. The parameters wlower and wupper correspond to the relative importance of lower and upper bounds. Here we set wlower=0.7, wupper=0.3 and threshold=0.0002.

4.If convergence criterion is met, i.e. cluster centers are same to those in previous iteration, then stop; else go to step2.

### 3.3.1 MapReduce function

The input to the Map function includes two parts, the input document dataset and the centroids got from PSO or from the last iteration are stored in centers directory in HDFS. First the input document dataset is split as a series of <key, value> pairs in each node where key is the CentriodId from MR-PSO module and value is the list of document Vectors. At each Mapper calculate the similarity between each document vector and the cluster centroids using Cosine similarity and compares distance between document vector and the cluster centroids to all the clusters and assign the document vector to the lower bound or upperbound of cluster centroids This is repeated for all the document vectors. A list of <key, value> pairs with ClusterID as the key and cluster contents as value is given to reduce function. The reduce function recalculate the new center value and update it. The pseudo code for the MapReduce job to perform RK-Means clustering using optimal centroids is given in **Figure.4**.

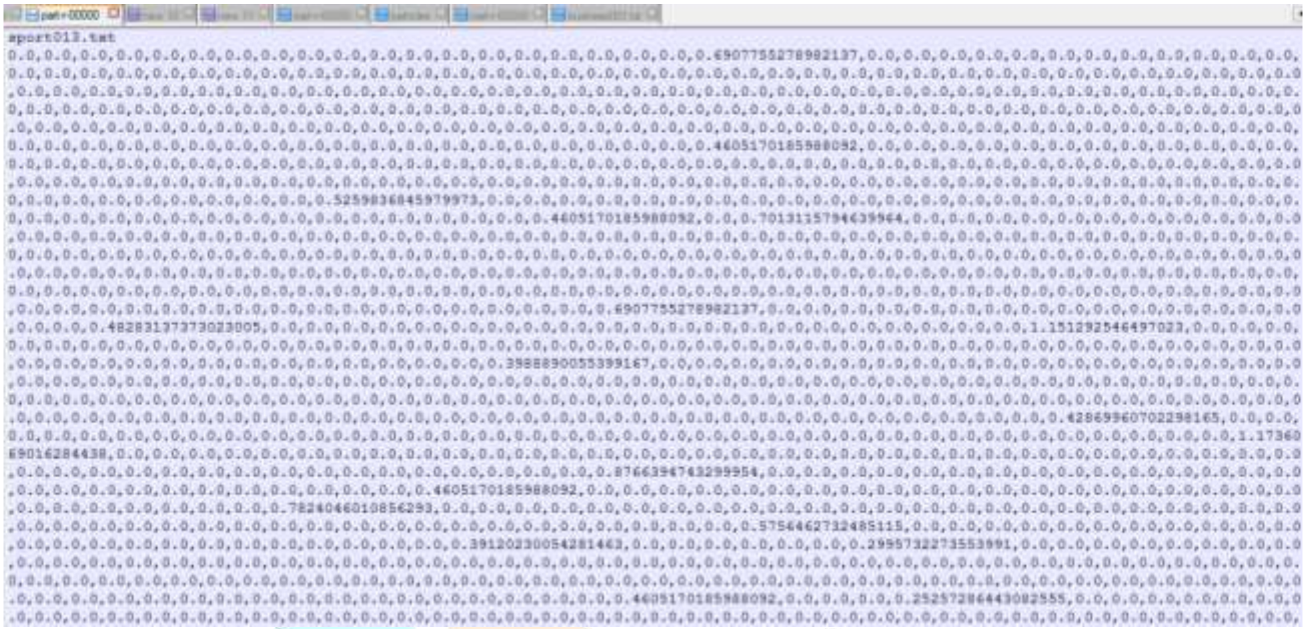


```
Function Map (key:CentriodID , value:DocID,vectorList)
 Find the nearest cluster pair
 for each docID in documents
      for each centroidID in centers
              find the distance between cluster centroid and vectors and
               take the difference between nearest cluster centroid and neighbour cluster centroid
  if the difference < some threshold value then
                set lower=false //it is a part of upper bound
                new_key=clusterId
                new_val=DocId,vector
                 emit(new_key, new_val);
  else
               set lower=true //it is a part of lower bound
               new_key=nearestclusterId
               new_val=DocId,vector
               emit(new_key, new_val);
     end for
 end function
Function Reduce (key:clusterID, valList:DocId,vector)
  #Update the centroids for each cluster
For each cluster xi
              Re-compute new center according to the equations.
              emit(clusterID, DocId)
end function
```

Fig. 4 The pseudo code for the MapReduce job to perform *R*K-Means clustering.

## IV. RESULT AND ANALYSIS

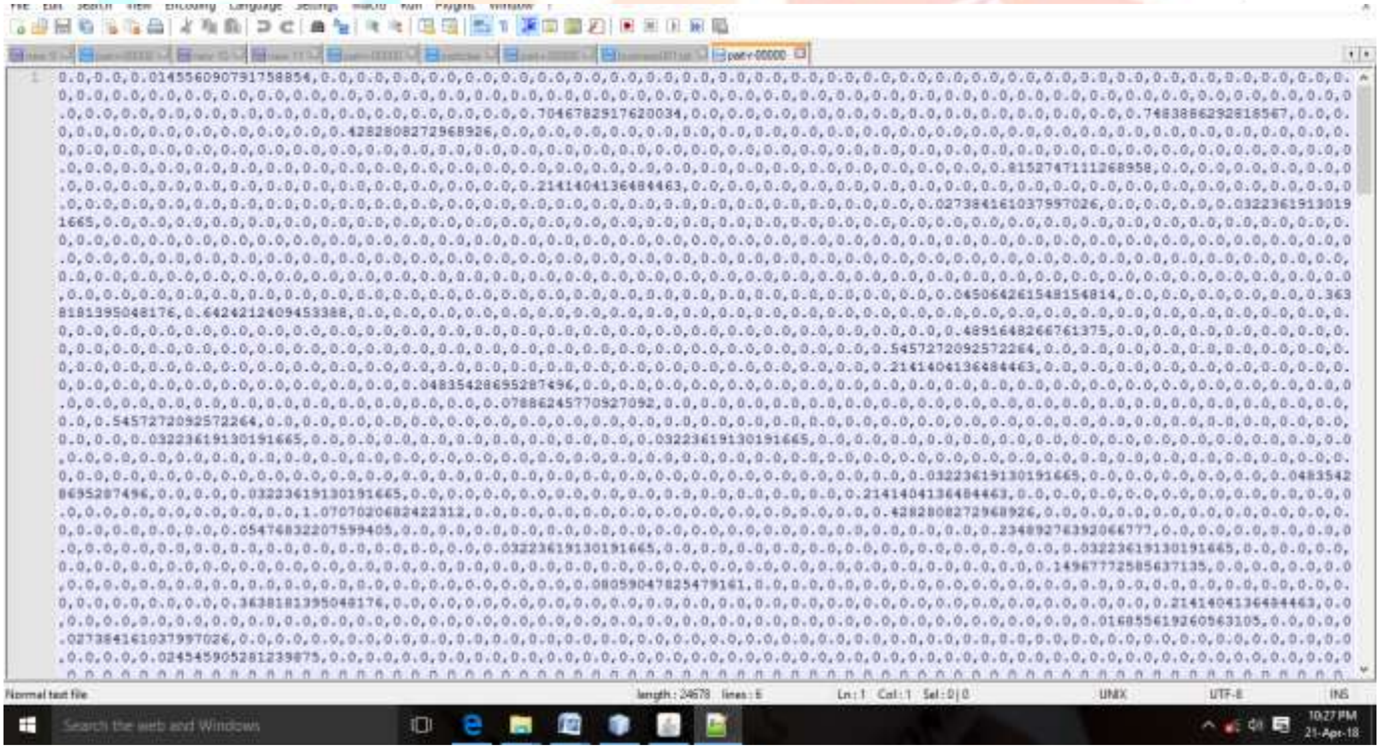
The proposed system is executed using Net Beans 8.0.2 IDE and Java language to develop our system and also use the Hadoop framework for distributed applications. The input to the MapReduce functions is a set of key-value pairs (key, value) and they are worked together simultaneously. The preprocessed data is given to the system then it finds the tfidf of each of the documents as shown in **Figure. 5.**



```
sport013.txt
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.6907755278982137,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0
.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.4605170185988092,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.5259836845979973,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.4605170185988092,0.0,0.0,0.7013115794639964,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.6907755278982137,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
,0.0,0.0,0.0,0.0,0.48283137373023005,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.151292546497023,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.3988890055399167,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0
.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.42869960702298165,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.17360
69016284438,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.8766394743299954,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.4605170185988092,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.7824046010856293,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.5754462732485115,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.39120230054281463,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.2995732273553991,0.0,0.0,0.0,0.0,0.0,0.0,0.0
,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0
.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.4605170185988092,0.0,0.0,0.25257286443082555,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0
.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
```

Fig.5 Document Vector Model (tfidf value).

Next MR-PSO takes this input and generates the optimal centriod value and finally it generates the cluster structure using MR-RoughKmeans algorithm. The PSO output is shown in **Figure.6**. And final output clustering is as shown in **Figure.7**.


File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
part-00000
1

```
0.0,0.0,0.014556090791758854,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0
.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.7046782917620034,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.7483886292818567,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.4282808272968926,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0
.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.8152747111268958,0.0,0.0,0.0,0.0,0.0,0.0,0
.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.2141404136484463,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.027384161037997026,0.0,0.0,0.0,0.0,0.0,0.0322361913019
1665,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0
.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.045064261548154814,0.0,0.0,0.0,0.0,0.0,0.0,0.363
8181395048176,0.6424212409453388,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.4891648266761375,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.5457272092572264,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.2141404136484463,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.04835428695287496,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.07886245770927092,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.5457272092572264,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.03223619130191665,0.0,0.0,0.0,0.0,0.0,0.0,0.03223619130191665,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.03223619130191665,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0483542
8695287496,0.0,0.0,0.0,0.03223619130191665,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.2141404136484463,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0
.0,0.0,0.0,0.0,0.0,0.0,1.0707020682422312,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.4282808272968926,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.05474832207599405,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.23489276392066777,0.0,0.0,0.0,0.0,0.0,0.0,0
.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.03223619130191665,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.03223619130191665,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.14967772585637135,0.0,0.0,0.0,0.0,0.0
,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.08059047825479161,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.
0,0.0,0.0,0.0,0.0,0.3638181395048176,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.2141404136484463,0.0
,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.016855619260563105,0.0,0.0,0
.027384161037997026,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
.0,0.0,0.0,0.0,0.024545905281239875,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
```

```
Normal text file          length : 24678    lines : 6          Ln : 1    Col : 1    Sel : 0 | 0          UNIX          UTF-8          INS
```

Fig.6 The Output of PSO.

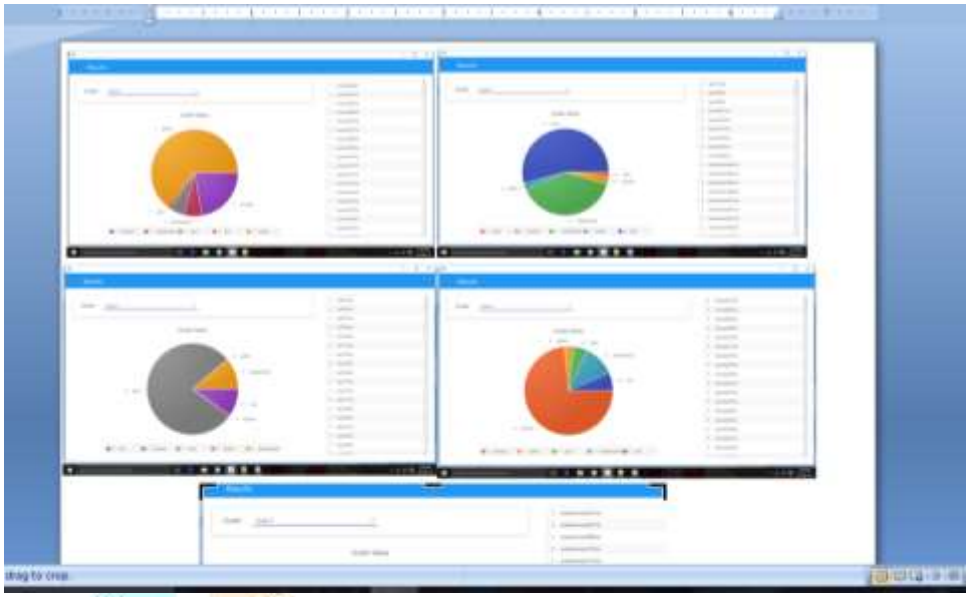


Fig.7 Final Output-Clustering No. and Corresponding Document Name.

## V. CONCLUSION

In this paper, design and implementation of a PSO and RoughKMeans clustering (MR-PRKMeans) algorithm using MapReduce framework was proposed. The PRKMeans clustering algorithm is an effective method for document clustering; however, it takes a long time to process large data sets. Therefore, MR-PRKMeans was proposed to overcome the inefficiency of PRKMeans for big data sets. The proposed method can efficiently be parallelized with MapReduce to process very large data sets. In MR-PRKMeans the clustering task that is formulated by RoughKMeans algorithm utilizes the best centroids generated by PSO.

## VI. ACKNOWLEDGMENT

We would like to thank all my friends and staffs in P.A College of engineering, Mangalore for their feedback and support.

## REFERENCES

[1] “Distributed Document Clustering Analysis Based on a Hybrid Method” J.E. Judith, J. Jayakumari. China Communications - February 2017. Received: Mar. 19, 2015 Revised: Sep. 18, 2016.

[2] “Design and Implement of Distributed Document Clustering Based on MapReduce” Jian Wan, Wenming Yu1, and Xianghua Xu. Proceedings of the Second Symposium International Computer Science and Computational Technology(ISCSCT ’09) Huangshan, P. R. China, 26-28,Dec. 2009, pp. 278-280. ISBN: 978-952-5726-07.

[3] M.F. Porter, “An algorithm for suffix stripping”, Program electronic library and information systems, vol. 14, Issue 3, pp.130 – 137, 1980.

[4] “Hadoop: A New Approach For Document Clustering”, Y.K. Patil and Prof. V.S. Nandedkar. International Journal of Advanced Research in IT and Engineering. ISSN: 2278-6244, Impact Factor: 4.054.

[5] “Document Clustering using Particle Swarm Optimization” Xiaohui Cui, Thomas E. Potok, Paul Palathingal.@2005IEEE.

[6] “Document Clustering Analysis Based on Hybrid PSO+K-means Algorithm” Xiaohui Cui and Thomas E. Potok.Journal of Computer Sciences (Special Issue): 27-33, 2005 ISSN 1549-3636.

[7] “Parallel PSO Using MapReduce”, Andrew W. McNabb, Christopher K. Monson, and Kevin D. Seppi. 1-4244-1340-0/07$25.00©2007 IEEE.

[8] “A Survey on K-mean Clustering and Particle Swarm Optimization”- Pritesh Vora, Bhavesh Oza. International Journal of Science and Modern Engineering (IJISME) ISSN: 2319-6386, Volume-1, Issue-3, February 2013.

[9] “Parallel Particle Swarm Optimization Clustering Algorithm based on MapReduce Methodology” Ibrahim Aljarah and Simone A. Ludwig. 978-1-4673-4769-3/12/$31.00 ©2012 IEEE.

[10] “Rough Set Approach in Machine Learning: A Review” Prerna Mahajan,Rekha Kandwal ,Ritu Vijay. International Journal of Computer Applications (0975 – 8887) Volume 56– No.10, October 2012.

[11] “Some Refinements of Rough K-Means Clustering” Georg Peters. Pattern Recognition 39 (2006) 1481–1491.

[12] “Comparative Study of K-Means, Pam and Rough K-Means Algorithms Using Cancer Datasets” Parvesh Kumar and Siri Krishan Wasan. 2009 International Symposium on Computing, Communication, and Control (ISCCC 2009) Proc .of CSIT vol.1 (2011) © (2011) IACSIT Press, Singapore.