COMMUNICATIONS IN SCIENCE AND TECHNOLOGY
Homepage: cst.kipmi.or.id

Comparison of distributed K-means and distributed fuzzy C-means algorithms for text clustering

I Made Artha Agastya, Teguh Bharata Adji, Noor Akhmad Setiawan*

Department of Electrical Engineering and Information Technology, Faculty of Engineering, Gadjah Mada University

Article history:
Received: 31 March 2017 / Received in revised form: 6 May 2017 / Accepted: 8 May 2017

Abstract

Text clustering has been developed in distributed system due to increasing data. The popular algorithms like K-Means (KM) and Fuzzy C-Means (FCM) are combined with Map Reduce algorithm in Hadoop Environment to be distributable and parallelizable. The problem is performance comparison between Distributed KM (DKM) and Distributed FCM (DFCM) that uses Tanimoto Distance Measure (TDM) has not been studied yet. It is important because TDM's characteristics are scale invariant while allowing discrimination collinear vectors. This work compared the combination of TDM with DKM (DKM-T) and TDM with DFCM (DFCM-T) to acquire performance of both algorithms. The result shows that DFCM-T has better intra-cluster and inter-cluster densities than those of DKM-T. Moreover, DFCM-T has lower processing time than that of DKM-T when total nodes used are 4 and 8. DFCM-T and DKM-T can perform clustering of 1,400,000 text files in 16.18 and 9.74 minutes but the preprocessing times take hours to complete.

Keywords: K-Means; FCM; Tanimoto Distance; MapReduce; Hadoop

1. Introduction

Data Mining (DM) [1] is a branch of Artificial Intelligence (AI) that focuses on retrieving information or knowledge from bunch of data. DM development is influenced by data that are produced by device around humans. Data can be structured, semi structured, or unstructured. Structured data are row-column data e.g. CSV and XLS data. Semi structured data can be XML and JSON data. Unstructured data consist of picture, video, music, text, etc. All of data types continue to grow and become too large and too fast to be processed using traditional methods.

Another popular term in information technology is Big Data. Big Data [2] has at least one of these characteristics i.e. Volume, Velocity, and Variety (3V). The term Volume means data from those devices become really big and make most computers cannot handle them. On the other hand, the data are produced in a very fast real time which leads to the term Velocity. As has been mentioned, the produced data are varied and thus it refers to Variety. All 3V's problems can be solved using Big Data Technologies which are strongly related to Hadoop Framework and MapReduce Algorithm.

Most of data in the world is text and it will be wasted if the vast amount data is not processed to gain knowledge or information. Hence, Hadoop is developed in order to handle massive text data in efficient and effective ways. Hadoop [3] strong aspects are scalability and affordability. The data is divided in chunk and distributed in cluster. Then it is processed in parallel ways so the process becomes faster than single computer processing.

Text clustering [4] is one of text mining categories which is used extensively for clustering news or document. Most algorithms which are used are K-Means and Fuzzy C-Means (FCM). Both algorithms are not suitable to process massive news and document. To improve the algorithm capabilities, K-Means and FCM are combined with MapReduce algorithm. The Distributed K-Means (DKM) and Distributed FCM (DFCM) are implemented in Hadoop platform and it was proven that it is effective for document and news clustering [5].

Comparison between K-Means and FCM was already performed in several data types. For example, there was comparison that used image data [6], intrusion data [7], or structured data [8]. Moreover comparison between DKM and DFCM has been performed for Wikipedia [5,9], Twitter [10] [11], and KDD Cup 1999 Data [12].

Most of those research works are using Euclidean Distance Measure (EDM) and Cosine Distance Measure (CDM) for measuring gap between instance and centroid. EDM is good for numerical data but it is bad for text data [11]. On the other hand, CDM is good for handling text data because of its scale invariant characteristic. Therefore, it is often used in text clustering or classification. Based on literature [13] CDM can miss relative distance between instance and centroid. Hence, Tanimoto Distance Measure (TDM) or Jaccard Distance Measure (JDM) is developed to capture both relative distance and degree between instance and centroid. Therefore, TDM [14] has both advantages of EDM and CDM which are scale

* Corresponding author.
Email: noorwewe@ugm.ac.id

invariant while allowing discrimination collinear vectors. In accordance with performance of distance measure, TDM [14,15,16,17] is one of best distance measure for clustering text. Because of several reasons stated before, it is important to understand the behavior of TDM in distributed environment especially in DKM and DFCM.

Based on literature review, the problem is the performance of DKM that uses TDM (DKM-T) and DFCM that uses TDM (DFCM-T) have not been revealed yet. Therefore, this research is conducted to gain knowledge about characteristics of DKM and FCM which apply Tanimoto Distance Measure.

This paper discusses four sections which are Introduction, Material and Method, Result and Discussion, and Conclusion.

2. Materials and Methods

In this section the dataset for clustering, applied methodologies, and tools are explained. The methods are tokenization, pruning, TF-IDF, DKM, DFCM, Tanimoto Distance Measure, inter-cluster density, and intra-cluster density.

2.1. Materials

In this research, the dataset [18] from Yahoo answers is used. Dataset consists of 1,400,000 questions and answers in CSV format. The size of CSV file is 780 MB. The dataset is available at the following link:

https://drive.google.com/open?id=0Bz8a_Dbh9Qhbfll6bVpmNUtUcFdjYmF2SEpmZUZUcVNiMUw1TWN6RDV3a0JHT3kxLVhVR2M

2.2. Tools

For preprocessing we used one laptop i5-3210M and 8 GB memory to overcome out of memory problems. To implement DKM and DFCM we used nine computers. One computer is Name Node and eight computers are Data Node. The specification of hardware and software are as follows:

A. Hardware for preprocessing
   1. One Laptop: 2.5 GHz Intel Core i5-3210M
   2. Memory: 8 GB DDR3
   3. OS: GNU/Linux Ubuntu version 14.04 LTS
   4. Hard Disk: 150 GB

B. Hardware for clustering
   1. Nine Computers: Intel Core 2 Duo 2.4 GHz.
   2. Memory: 2 GB DDR3
   3. OS: GNU/Linux Ubuntu version 14.04 LTS
   4. Hard Disk: 190 GB.

C. Software
   1. Hadoop version 2.6.2
   2. Openjdk-7-jdk
   3. Openssh-server
   4. Mahout version 0.12.2
   5. Python version 3.4.2

2.3. Methods

All fundamental theories are explained briefly in every sub section. The method is based on fundamental theories and is designed to meet the purpose of the research.

2.3.1. Tokenization

Tokenization is a technique to remove punctuation mark or white space so the word becomes independent as shown in Fig. 1.

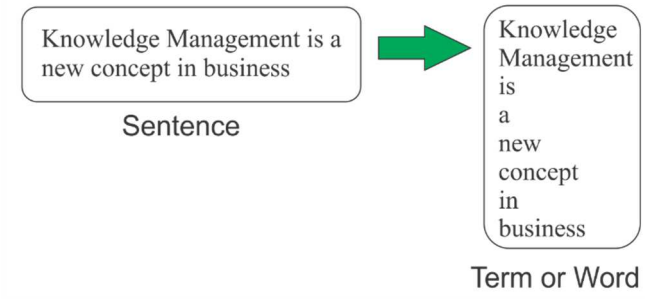

Fig. 1. Tokenization illustration

2.3.2. Pruning

Pruning is a method to remove very high frequency or very low frequency word. Using example in Fig. 1, frequency of “a” is 100 and frequency of “business” is 1 but other examples have frequencies in between 3 and 10. We set pruning rules that remove the most and the least occurring word. Hence, the “a” and “business” words are neglected.

2.3.3. TF-IDF

Term Frequency (TF) is frequency of occurring word or term in a document. TF will be high if the frequency of occurring word in the document is high.

Inverse Document Frequency (IDF) is combination which diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely. A little document frequency (df) indicates that the word is a significant term. TF-IDF (W) is calculated using both TF and IDF as shown in (1) and (2).

$$IDF_j = log\left({}^{D}/_{df_j}\right) \quad (1)$$

$$W_{ij} = TF_{ij} \times IDF_j \quad (2)$$

2.3.4. MapReduce

MapReduce [19] is a framework which is used for executing distributable and parallelizable algorithm. Sequential algorithm can become distributable algorithm if the algorithm is adjusted to MapReduce framework. MapReduce has two important tasks which are Map task and Reduce task. Map task converts a set of data to become another set of data which the original set of data is split into tuple as key and value pairs. Next, the Reduce task takes the output from map

task and uses it for input. Then the collected tuples is combined to make smaller set of tuples.

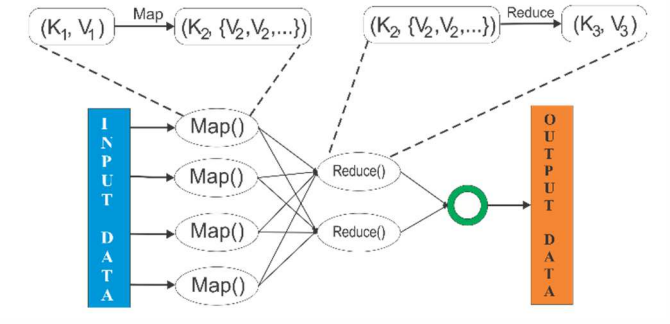

Fig. 2. Map reduce algorithm

As shown in Fig. 2, the tuple (K_1, V_1) is mapped to new tuple (K_2, {V_2, V_2, …}). Then new tuple (K_2, {V_2, V_2, …}) is reduced to tuple (K_3, V_3). The desired algorithm (e.g. K-Means or FCM) need to be modified to behave like MapReduce Algorithm.

2.3.5. Distributed Fuzzy C-Means (DFCM)

Fuzzy c-means (FCM) is a data clustering technique in which a dataset is grouped into n clusters with every data point in the dataset belonging to every cluster to a certain degree. For example, a certain data point that lies close to the center of a cluster will have a high degree of belonging or membership to that cluster. Meanwhile, another data point that lies far away from the center of a cluster will have a low degree of belonging or membership to that cluster.

FCM algorithm objective is reducing value of J or Cost function as (3). N is represented as total member of cluster. c represents cluster index. x_j is position of data. v_j is centroid of cluster. u_{ij} is membership function.

$$J = \sum_{j=1}^{N}\sum_{i=1}^{c} u_{ij}^2 (x_j - v_i)^2 \tag{3}$$

The process of FCM algorithm is explained in the following steps:

1. Determine total cluster and initiate centroid of cluster v_i.
2. Calculate membership function u_{ij}

$$u_{ij} = \frac{1}{\sum_{k=1}^{c}\left(\frac{(x_j - v_i)}{(x_j - v_k)}\right)^2} \tag{4}$$

3. Calculate new centroid of cluster v_i

$$v_i = \frac{\sum_{j=1}^{N} u_{ij}^2 x_j}{\sum_{j=1}^{N} u_{ij}^2} \tag{5}$$

4. Test whether cluster is already convergent or not.
5. Update membership function u_{ij}

Distributed Fuzzy C-Means (DFCM) [13] is a combination of FCM and MapReduce algorithm. Hence, Map task is launched when calculating membership function u_{ij}. Next, Reduce task is applied when averaging membership functions to determine new centroid. The Map and Reduce task is the main task to gain paralleled or distributed process. DFCM is implemented in Mahout [20] which is scalable machine learning library for large dataset.

2.3.6. Distributed K-Means (DKM)

K-means is an unsupervised learning algorithm that can be used for clustering dataset. The steps are simple and easy and the objective is classifying a given data set through a certain number of clusters *c*. K-means Clustering Algorithm is composed of the following steps:

1. Choose *c* initial centroids.
2. Assign each instance to the group that has the closest centroid.
3. When all instances have been assigned, recalculate the positions of the *c* centroids based on means of instances position.
4. Repeat Steps 2 and 3 until the centroids no longer move.

Similar to DFCM, the algorithm of DKM [13] is a combination of K-Means and MapReduce algorithm. The Map task is conducted while assigning all instance with closest centroid. Then Reduce task is performed when calculating average of every cluster member to gain new centroid. As well as DFCM algorithm, DKM algorithm is conducted in Mahout [20].

2.3.7. Tanimoto Distance

Cosine Distance Measure does not capture relative distance between two instances but only calculates cosine value between two vectors. On the other hand, Tanimoto Distance Measure can capture the relative distance and angle between vectors. Both information can be used for distinguishing two vectors better than only one information. How to get Tanimoto Distance (R) is explained in (6) and (7). The respective vectors are *p* and *q*.

$$R = p_1 q_1 + p_2 q_2 + \cdots + p_n q_n \tag{6}$$

$$d = 1 - \frac{R}{\sqrt{({p_1}^2 + {p_2}^2 + \cdots + {p_n}^2)} + \sqrt{({q_1}^2 + {q_2}^2 + \cdots + {q_n}^2)} - R} \tag{7}$$

2.3.8. Intra-cluster Density

Intra-cluster Density (Di) is density in the cluster that is calculated using distance between instance and centroid in cluster as represented in (8). Distance between instance and centroid in cluster is represented as *di*. Total instance is represented as *N*. The greater the intra-cluster density, the more compact the cluster.

$$Di = \frac{\frac{\sum_{j=1}^{N} di_j}{N} - min(di)}{max(di) - min(di)} \tag{8}$$

2.3.9. Inter-cluster Density

Inter-cluster Density (Da) is density between clusters which is calculated using distance between cluster centroid as

shown in (9). Distance between cluster centroids is represented as *da*. Total cluster is represented as *M*. The smaller the inter-cluster density, the more distinct the clusters.

$$Da = \frac{\frac{\sum_{j=1}^{M}\sum_{k=1}^{M} da_{jk}}{M^2} - min(da)}{max(da) - min(da)} \quad (9)$$

2.3.10. Work Flow

Work flow of comparing both algorithms is started by converting the dataset in CSV format to 1,400,000 text files and then convert the text files to sequential format. Sequencing files format is mandatory if we want to process a dataset in Hadoop environment. After the file is in sequential format, it is converted to vector.

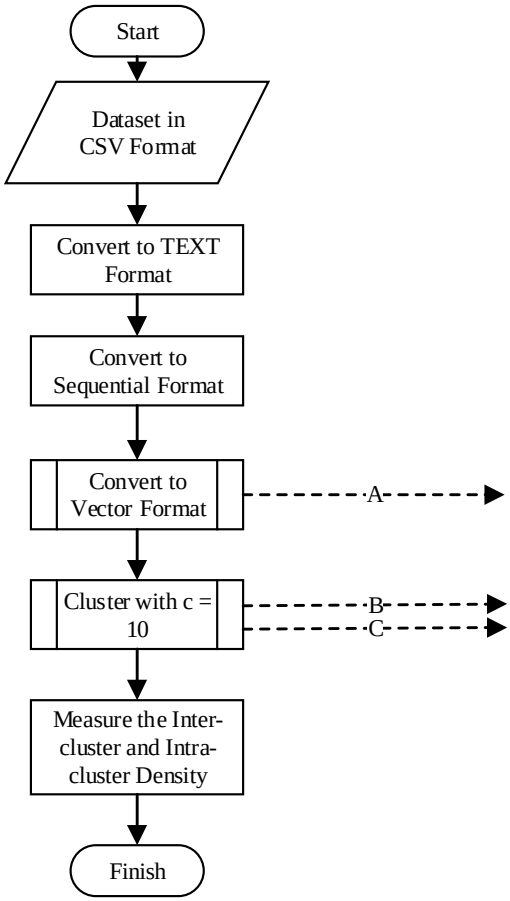

Fig. 3. General process of comparing DKM-T and DFCM-T

When sequential file is converted to vector, the preprocessing is later conducted as follows:

1. Tokenization
   Convert the document to tokens or terms.
2. Term Frequency (TF)
   Calculate frequency of terms.
3. Pruning
   Remove 1 % of highest frequency words.
4. Term Frequency-Inverse Document Frequency (TF-IDF)
   Weighting the terms, the term that rarely appear in all documents gains bigger weight than the term that often appear in all documents.

After that the vector files are processed in two testing scenarios which are:

1. All dataset is clustered using ten clusters (c = 10) in local, 2 nodes, 4 nodes, and 8 nodes. Then the computational time is recorded. This first scenario process is shown in Fig. 4.
2. All dataset, half of dataset, a quarter of dataset, and 10% of dataset is clustered using ten clusters (c = 10) in 8 nodes. Then the computational time is recorded. This second scenario process is shown in Fig. 5.

After the clusters are available, all clusters are evaluated with inter-cluster density and intra-cluster density. We choose ten clusters (c = 10) because the dataset itself has ten classes. The dataset [18] was used by Zhang et al. for solving text classification problems. The general process of comparing both algorithms is presented in Fig. 3.

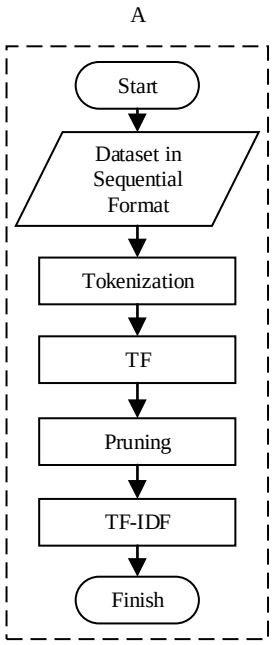

Fig. 4. Preprocessing

To get fair results, both algorithms are conducted in same initial cluster. Using same initial cluster leads both cluster algorithms to search convergence solution in same space or region. There are four initial clusters namely initial cluster for 10% dataset, 25% dataset, 50% dataset, and 100% dataset.

Example of Mahout command line that is used to perform both algorithms is as follows:

a) DKM-T

```
hduser@master:~/mahout$ bin/mahout kmeans -i yahoo-answers-full-vectors-default/tfidf-vectors/ -c yahoo-answers-initial-clusters -o yahoo-answers-full-vectors-default-kmeans-clusters-tanimoto -dm org.apache.mahout.common.distance.TanimotoDistanceMeasure -cd 0.1 -x 20 -cl -ow
```

b) DFCM-T

```
hduser@master:~/mahout$ bin/mahout fkmeans -i yahoo-answers-full-vectors-default/tfidf-vectors/ -c yahoo-answers-initial-clusters -o yahoo-answers-full-vectors-default-fcm-clusters-tanimoto -dm
```

```
org.apache.mahout.common.distance.TanimotoDistanceMeasu
re -cd 0.1 -m 2 -x 20 -ow -cl
```

B

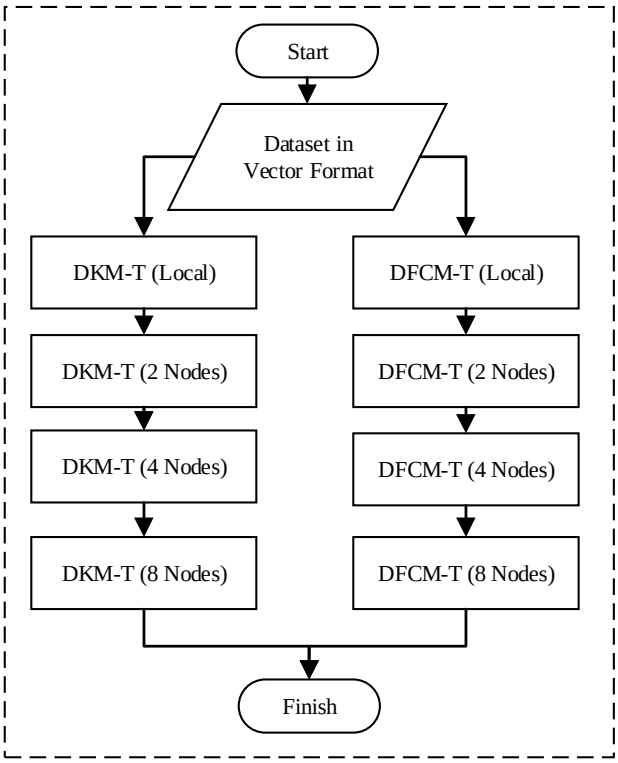

Fig. 5. The 1st Scenario of DKM-T and DFCM-T

C

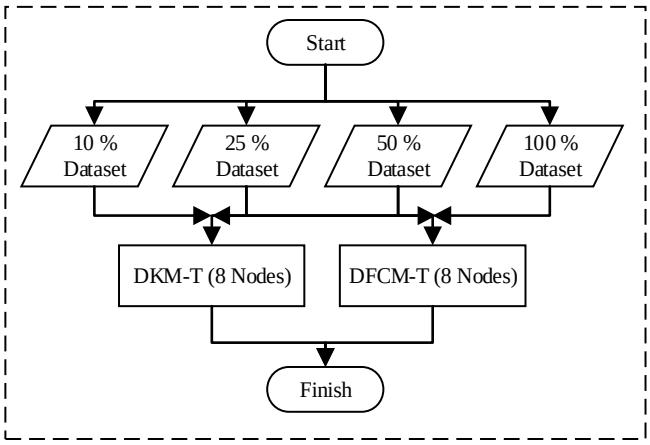

Fig. 6. The 2nd scenario DKM-T and DFCM-T

3. Results and Discussion

In this section, the result will be presented in tables and figures and will be explained in four sections namely preprocessing, first scenario, second scenario, and cluster evaluation.

3.1. Preprocessing

Before 1,400,000 text data can be processed, it needs to be converted to sequential files. It takes hours to convert bunch of text files to sequence files as shown in Table 1. This preprocessing is the main problems of DKM-T and DFCM-T. The cause of high processing time is cumulative time to read and extract text data one by one and then convert it to sequential files.

```
hduser@master:~/mahout$ hadoop fs -put yahoo-answers-
full /user/hduser/

Exception in thread "main" java.lang.OutOfMemoryError:
Java heap space
        at java.net.URI.toString(URI.java:1917)
        at java.net.URI.<init>(URI.java:749)
        at
org.apache.hadoop.fs.Path.initialize(Path.java:203)
        at
org.apache.hadoop.fs.Path.<init>(Path.java:197)
        at
org.apache.hadoop.fs.RawLocalFileSystem.listStatus(RawLo
calFileSystem.java:396)
        at
org.apache.hadoop.fs.FileSystem.listStatus(FileSystem.ja
va:1485)
        at
org.apache.hadoop.fs.FileSystem.listStatus(FileSystem.ja
va:1525)
        at
org.apache.hadoop.fs.ChecksumFileSystem.listStatus(Check
sumFileSystem.java:570)
        at
... ...
```

Fig. 7. Error message when uploading text data to HDFS

All text data is converted to sequence file using laptop with 8 GB memory because the Hadoop cluster cannot perform the process due to error when uploading files to HDFS. The error happens since the quantity of files is beyond memory capability of Hadoop Cluster in which each computer only has 2 GB memory. The error message is shown in Fig. 7.

Table 1. Processing time for converting text files to sequence files

Dataset	Total Input Texts (each)	Times (Minutes)	Total Output Sequence File (each)
10% dataset	175,000	38.1	2
25% dataset	350,000	102.9	3
50% dataset	700,000	242.7	6
100% dataset	1,400,000	443.9	11

Replication	Block Size	Name
0	0 B	df-count
3	128 MB	dictionary.file-0
3	128 MB	frequency.file-0
0	0 B	tf-vectors
0	0 B	tfidf-vectors
0	0 B	tokenized-documents
0	0 B	wordcount

Fig. 8. File vector in HDFS

After sequential files are provided, it needs to be processed using tokenization, TF, pruning, and TF-IDF. Every preprocessing step (except pruning) is made into a new file

and is separated in different folder as shown in Fig. 8. The Fig. 8 shows HDFS folder view after the vector files is uploaded in HDFS. The replication value means total data replication across nodes to prevent data failed to be processed in result of some data loss. The block size means the maximum data chunk size. If the data is more than 128 MB then the data must be split to several chunk. After all processes are done then the sequential files become vectors and are ready to use for clustering. Based on Table 2, preprocessing times are much smaller than converting text files to sequence files. This happens because the input sequential files are 2, 3, 6, and 11 files. It is much lesser than 175,000, 350,000, 750,000, 1,400,000 text files. The total file processed is very influential on the computation time.

Table 2. Processing time for converting sequences files to vectors

Dataset	Total Input Sequence File (each)	Times (Minutes)
10% dataset	2	1.349
25% dataset	3	3.093
50% dataset	6	7.290
100% dataset	11	24.347

3.2. First Scenario

We also examine the clustering computation time using local, 2 nodes, 4 nodes, and 8 nodes for 100% dataset. DFCM-T becomes faster than DKM-T when nodes reaches 4. This indicates that DFCM-T has better scalability than that of DKM. Moreover, the DFCM-T's computation time decreases gradually but DKM-T's computation time decreases fast in 2 nodes and then decreases slowly in 4 nodes and 8 nodes as shown in Table 3.

Table 3. Comparison of processing time for different total nodes

Clusters	DKM-T (Minutes)	DFCM-T (Minutes)
Local	54.85	26.17
2 Nodes	16.67	17.31
4 Nodes	16.31	13.40
8 Nodes	16.18	9.74

3.3. Second Scenario

Table 4. Comparison of processing time for different dataset size

Dataset	DKM-T (Minutes)	DFCM-T (Minutes)
10% dataset	7.14	3.12
25% dataset	10.02	4.60
50% dataset	11.68	6.25
100% dataset	16.18	9.74

This scenario is implemented in 8 nodes and uses difference data size. For both DKM-T and DFCM-T have similar trend. It is normal when the dataset size increases, the computational times also increases as presented in Table 4.

3.4. Cluster Evaluation

Based on information in Table 5, DFCM-T has greater average intra-cluster density than DKM-T. It means that DFCM-T has better cluster distribution than DKM-T. The distribution of cluster indicates the similarity of instances in the cluster. The smaller distance between instances or the more compact of cluster, the more possible instances are in the right cluster.

Table 5. Comparison of intra-cluster density

Cluster	DKM-T	DFCM-T
1	0.599	0.618
2	0.689	0.693
3	0.677	0.660
4	0.565	0.623
5	0.638	0.676
6	0.615	0.649
7	0.686	0.587
8	0.578	0.574
9	0.527	0.684
10	0.677	0.627
Average	0.625	0.639

As shown in Table 6, inter-cluster density of DFCM-T is smaller than DKM-T so DFCM-T's clusters are more separated than DKM-T's cluster. The separation between clusters indicates that the cluster is different from each other. So the possibilities that we get the right clusters are high.

DFCM-T is better than DKM-T because the instances seem to be related to each other. Therefore, the fuzzy approach is more suitable than crisp approach.

Table 6. Comparison of inter-cluster density

Density	DKM-T	DFCM-T
Inter-cluster	0.312	0.205

4. Conclusion

DKM-T's characteristic is much reduced computational time in 2 nodes but almost stucks in more than two nodes. This is because the DKM-T scalability becomes saturated when it uses 2 nodes. In terms of cluster quality, DKM-T is worse than DFCM-T. This is because the instances seem to be related to each other. Therefore, fuzzy approach gives better result than crisp approach. Both DKM-T and DFCM-T have problem in preprocessing. Both algorithms preprocessing time are very high because processing a lot of small text files is really exhausting. Even if we want to process in Hadoop Environment, we still need to upload files to HDFS. Moreover, uploading a lot of small files in HDFS is not possible due to out of memory problems. In order to overcome the problems we need to convert CSV file that contains

question and answer to sequential file directly. Hence, the file can be processed in HDFS or directly in local system. In fact the DKM-T becomes saturated when it uses 2 nodes which must be studied further.

References

1. X. Wu, X. Zhu, G. Wu, and W. Ding, *Data mining with big data*, IEEE Trans. Knowl. Data Eng. 26 (2014) 97–107.
2. A. Gandomi and M. Haider, *Beyond the hype: Big data concepts, methods, and analytics*, Int. J. Inf. Manage. 35 (2015) 137–144.
3. T. White, *Hadoop : The Definitive Guide*, 4th ed. Sebastopol, CA: O'Reilly Media, 2015.
4. S. Sathya and N. Rajendran, *A review on text mining techniques*, Int. J. Comp. Sci. Trend. Technol. 3 (2015) 274–283.
5. R. C. E. Rui, *Using Mahout for clustering wikipedia's latest articles: A comparison between K-means and Fuzzy C-means in the cloud*, *3rd IEEE Int. Conf. Cloud Comput. Technol. Sci., Athens, Greece, 2011, pp. 565–569.*
6. S. Madhukumar and N. Santhiyakumari, *Evaluation of K-means and Fuzzy C-means segmentation on MR images of brain*, Egypt. J. Radiol. Nucl. Med. 46 (2015) 475–479.
7. S. K. Sahu and S. K. Jena, *A study of K-Means and C-Means clustering algorithms for intrusion detection product development*, Int. J. Innov. Manag. Technol. 5 (2014) 207–213.
8. S. Panda, S. Sahu, P. Jena, and S. Chattopadhyay, *Comparing Fuzzy-C means and K-means clustering techniques: A comprehensive study*, Adv. Intell. Soft Comput. 166 (2012) 451–460.
9. L. Sahu and B. R. Mohan, *An improved K-means algorithm using modified cosine distance measure for document clustering using Mahout with Hadoop, 9th International Conference on Industrial and Information Systems (ICIIS), Gwalior, India, 2014, pp. 1–5.*
10. E. Jain and S. K. Jain, *Using Mahout for clustering similar twitter users: Performance evaluation of K-means and its comparison with fuzzy K-means, Proc.-5th IEEE Int. Conf. Comput. Commun. Technol. ICCCT, Allahabad, India, 2015, pp. 29–33.*
11. E. Jain and S. K. Jain, *Categorizing twitter users on the basis of their interests using Hadoop/Mahout platform, 9th International Conference on Industrial and Information Systems (ICIIS), Gwalior, India, 2014, pp. 1–5.*
12. P. M. D. Avila et al., *Comparing K-means and mean shift algorithms performance using Mahout in a private cloud environment*, J. Commun. Comput. 11 (2014) 45–51.
13. S. Owen, R. Anil, T. Dunning, and E. Friedman, *Mahout in action*, 1st ed. New York, US: Manning Publications, 2011.
14. J. Ghosh and A. Strehl, Similarity-based text clustering: *A comparative study*, Group. Multidimensional Data, 2 (2006) 73–97.
15. A. Huang, *Similarity measures for text document clustering, Proc. New Zeal. Comput. Sci. Res. Student Conf., Christchurch, New Zealand, 2008, pp. 49–56,.*
16. A. Strehl, E. Strehl, J. Ghosh, and R. Mooney, *Impact of similarity measures on web-page clustering, Workshop on Artificial Intelligence for Web Search, Texas, USA, 2000, pp. 58–64.*
17. A. Rangrej, S. Kulkarni, and A. V. Tendulkar, *Comparative study of clustering techniques for short text documents, Proceedings of the 20th international conference companion on World Wide Web, Hyderabad, India, 2011, pp. 111–112.*
18. X. Zhang, J. Zhao, and Y. LeCun, *Character-level convolutional networks for text classification*, Comput. Res. Repositor. 1509.01626 (2016) 1-9.
19. B. Lublinsky, K. T. Smith, and A. Yakubovich, *Professional: Hadoop solutions.* Indiana, CA: John Wiley & Sons, 2013.
20. Apache, *Apache Mahout: Scalable Machine Learning and Data Mining. [Online]. Available: http://mahout.apache.org/.* [Accessed: 20-Mar-2017].