Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

DOCUMENT CLUSTERING USING SPARK

A PROJECT REPORT

Submitted by

17BCE2408 – Rachit Jain

17BCE2388 – Sweta Kumari

CSE4001 Parallel and Distributed Computing

Slot – L43+ L44

Computer Science and Engineering

JUNE 2020

Under the guidance of

Prof. BALAMURGAN R

TABLE OF CONTENTS

1.ABSTRACT :

We, with regards to this topic have tried to collect and understand all the existing techniques and methods. After in-depth analysis of the research papers we found some gaps that were not yet resolved. Our main aim in this survey is to point out those issues and gaps. We have listed below the summaries of the research papers followed by conclusion and discrepancies found. Finally we have outlined briefly solution to the problems and gaps found in the existing models. We have also, in short, defined the algorithm suited to find the optimum number of clusters for the given set of pdf documents. The optimum value of 'k' i.e number of clusters will be found using the elbow technique which we plan to implement in distributed system on Spark.

Keywords

Spark, Distributed System, Map-Reduce, Document Clustering, Kmeans, Spark, HDFS, YARN,

Elbow technique, Big Data, Eucleadian distance.

2.INTRODUCTION

Spark is an open source application of MapReduce .The Apache Spark framework develops open-source software for trusted, multiplicative, distributed computation. The Apache Spark services the distributed processing of big data sets throughout clusters of systems using a basic programming code. Because the machine resource management of a cluster is reactive to the number of cluster nodes, it is necessary to make the managing easy. To resolve the hardness of machine management, answer is virtualization, which is used to for simple configuration, application, scheduling and its efficiency in resource usage . Current virtual machine techs facilitate a one physical server to be distributed into several virtual resource boxes, each supplying a powerful, mature, and isolated implementation environment for jobs .

Big data has received high reignition in recent times due to the huge amount of data generated on regular basis. How to compute, send, and store these huge data is a standard hard job which will bring great effect on the native architectures and techniques of computation, networking and storage .

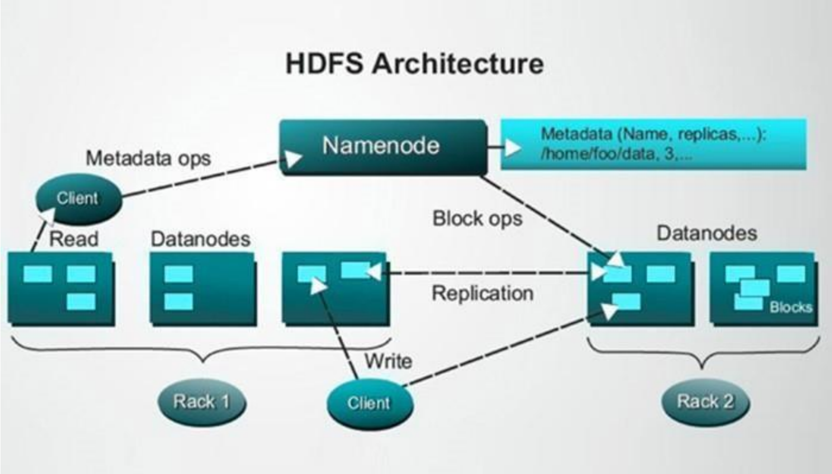

Figure 1: Architecture of HDFS

There are many methods available to evaluate big data on learning performances. Many times, big data sets, consist of some group similar points and it is important to nd these cluster in order to find trends in data [10]and [16]. One of the most popular clustering methods is k-means. It divides the data into clusters by minimizing a value between the samples and the centroid of the clusters. Manhattan distance is a simpler measure also for Euclidean space. Euclidean distance is a similarity criterion mainly used for samples in Euclidean space. Cosine distance and Jaccard is mostly applied for documents clustering[7].

Tools Used :

Apache Spark is an integrated computing engine and a set of libraries for parallel data processing on computer clusters. It is the most actively developed open source engine for this task. Making it a necessary tool for any developer or data scientist interested in Big Data. Spark supports the use of several widely used programming languages (Python, Java, Scala, and R),and it including libraries for various tasks ranging from SQL to streaming and machine learning, and from laptops to thousands of servers. It runs anywhere in the cluster. This makes Big Data Processing an easy system to start and scale on an incredibly large scale.

The Google Cloud Platform is a provider of computing resources for the deployment and operation of applications on the web. Its specialty is providing individuals and enterprises a place to build and run software, and it uses the web to connect with users of that software. Think of thousands of websites working on a network of ”hyperscale” (very large, but very divisible) data centers, and you’ll get the basic idea. It is a suite of public cloud computing services offered by Google. The platform includes many hosted services for compute, storage and application development that run on Google hardware.

3.Literature Review

Sr. No.	TITLE	JOURNAL NAME AND DATES	AUTHORS	KEY CONCEPTS	ADVANTAGES AND DISADVANTAGES, FUTURE SCOPE
1)	Parallel K-Medoids Clustering Algorithm Based on Hadoop	2014 IEEE	Yaobin Jian, Jiongmin Zhang	The parallel KMedoids clustering algorithm HK-Medoids is proposed and has the related experimental verification. - The key-value pairs are passed as parameters to map function and written to the local disk periodically	-The growth rate of speedup decreases owing to the increased machine communication. And the speedup reach bottleneck easily when dealing with small-sizes dataset. For big-data, the algorithm HK-Medoids can obtain a linear speedup. -The future work will be focused on optimizing algorithm scheduling on Hadoop platform and selecting the initial cluster centers.
2)	MapReduce Algorithms for kmeans Clustering	2015 IEEE	Max Bodoia	-The algorithm presented is a version of the standard algorithm which chooses the initial means differently. This algorithm is designed to choose a set of initial means which are wellseparated from each other.	- distributed algorithms K-means and Kmeans++ performed much like their serial counterparts: Kmeans++ produced clusters with lower average and minimum error than KMEANS, but unfortunately sacrificed speed in the process.

3)	Continuous Clustering in Big Data Learning Analytics	2013 IEEE	Kannan Govindarajan , Thamarai Selvi Somasundaram , Vivekanandan S Kumar , Kinshuk	-This study discusses a method to continuously capture data from students' learning interactions. Then, it analyzes and clusters the data based on their individual performances in terms of accuracy, efficiency and quality by employing Particle Swarm Optimization (PSO) algorithm.	-can be used to develop cloud-based MapReduce framework with Hadoop. Hence, performing clustering in parallel environment.

4)	Clustering large datasets using K-means modifed inter and intra clustering (KM-I2C) in Hadoop	2017 Sreedhar et al. J Big Data	Chowdam Sreedhar , Nagulapally Kasiviswanath and Pakanti Chenna Reddy	-The first approach, K-Means Hadoop MapReduce (KM-HMR), focuses on the MapReduce implementation of standard K-means. - The second approach enhances the quality of clusters to produce clusters with maximum intracluster and minimum intercluster distances for large datasets.	-enhance the performance of map and reduce jobs to suit large datasets in future. The performance of Hadoop can be enhanced by using multilevel queues for the efficient scheduling of jobs suitable for large datasets.
5)	Strategies for Big Data Clustering	2014 IEEE	Olga Kurasova, Virginijus Marcinkevicius, Viktor Medvedev, Aurimas Rape ˇ cka, and Pavel Stefanovi ˇ c	-An overview of methods and technologies used for big data clustering is presented. Clustering is one of the important data mining issue especially for big data analysis, where large volume data should be grouped. -Here some clustering methods are described, great attention is paid to the k-means method and its modifications	-Big data clustering is done without using distributed systems, hence, in future tools like Hadoop can be used to enhance the performance of the algorithm.

6)	MapReduce Design of K-Means Clustering Algorithm	2013 IEEE	Prajesh P Anchalia, Anjan K Koundinya, Srinath N K	This paper discusses the implementation of the K-Means Clustering Algorithm over a distributed environment using ApacheTM Hadoop. The key to the implementation of the K-Means Algorithm is the design of the Mapper and Reducer routines.	-In future the number of iterations required to cluster data can be reduced by choosing the right set of initial set of centroids. Hence, research on the better way to choose centroids.
7) CHRISTIAN SOHLER.	Fuzzy K-mean Clustering in MapReduce on Cloud Based Hadoop	2014 IEEE	Dweepna Garg ,Khushboo Trivedi	This paper focuses on studying the performance of different datasets using Fuzzy K-means clustering in MapReduce on Hadoop.	-Optimize the Fuzzy Kmean clustering algorithm by considering the centroid point using certain methodology instead of choosing it randomly which could result in a cluster.

8)	K-means clustering in the cloud - a Mahout test	2011 IEEE	Rui Máximo Esteves, Chunming Rong, Rui Pais	-Mahout is a cloud computing approach to K-Means that runs on a Hadoop system. Both Mahout and Hadoop are free and open source. -Due to their inexpensive and scalable characteristics, these platforms can be a promising technology to solve data intensive problems which were not trivial in the past.	-The overhead is largely compensated as soon as the dataset grows. Results not consistent for the small datasets. Hence, work can be done in future to mould the algorithm to better perform for small datasets.
9)	Addressing Big Data Problem Using Hadoop and Map Reduce	2012 NUiCONE	Aditya B. Patel, Manashvi Birla, Ushma Nair	-This paper reports the experimental work on big data problem and its optimal solution using Hadoop cluster, Hadoop Distributed File System (HDFS) for storage and using parallel processing to process large data sets using Map Reduce programming framework.	Increasing the number of nodes reduces the execution times. However, for small files it can lead to an underutilization of each machine's resources. Future work can be done on: -bigger datasets; increase in the number of tasks; - increase in the number of reducers; - one machine with n cores vs n machines with 1 core

10)	StreamKM++: A Clustering Algorithm for Data Streams	2012 JEA	MARCEL R. ACKERMANN, MARCUS MARTENS, CHRISTOPH RAUPACH, KAMIL SWIERKOT, CHRISTIANE LAMMERSEN,	- To compute the small sample, they propose two new techniques. First, use an adaptive, nonuniform sampling approach similar to the k-MEANS++ seeding procedure to obtain small coresets from the data stream. The use of these coreset trees significantly speeds up the time necessary for the adaptive, nonuniform sampling	Advantage-They have shown that this algorithm is capable of efficiently clustering huge amounts of data in the data streaming model. Future Scope – when the number of clusters is not available in advance, the proposed approach can be extended for finding out the natural number of clusters present in the data set, in addition to computing the initial
11)	Streaming k-means approximation	2009 IEEE	Nir Ailon, Ragesh Jaiswa, Claire Monteleoni	-Derivation of an extremely simple pseudoapproximation batch algorithm for kmeans (based on the recent k-means++), in which the algorithm is allowed to output more than k centers, and a streaming clustering algorithm in which batch clustering algorithms are performed on small inputs (fitting in memory) and combined in a hierarchical manner	streaming methods achieve much lower cost than Online Lloyd's, for all settings of k, and lower cost than Batch Lloyd's for most settings of k. Future scope: -compatibility for bigger data sets

12)	k-means++: The Advantages of Careful Seeding	2007 ACM-SIAM	David Arthur and Sergei Vassilvitskii.	By augmenting kmeans with a simple, randomized seeding technique, we obtain an algorithm that is O(log k)-competitive with the optimal clustering.	Advantage - k-means++ consistently outperformed k-means, both by achieving a lower potential value, in some cases by several orders of magnitude, and also by completing faster. With the synthetic examples, the k-means method does not perform well, because the random seeding will inevitably merge clusters together, and the algorithm will never be able to split them apart.
13)	How Slow is the kMeans Method?	2006 Symposium	David Arthur, Sergei Vassilvitskii	-The k-means method is an old but popular clustering algorithm known for its observed speed and its simplicity. Until recently, however, no meaningful theoretical bounds were known on its running time. -In this paper, they have demonstrated that the worst-case	Future scope: A simple variant of kmeans where only one data point is reassigned each iteration. This variant has running time polynomial in n and the spread Δ, which we know is not true for standard k-means. Further research can be done on this

				running time of kmeans is superpolynomial by improving the best known lower bound	
14)	MapReduce for Data Intensive Scientific Analyses	2008 IEEE	Jaliya Ekanayake, Shrideep Pallickara, and Geoffrey Fox	-the goal of this paper was to present the experience in applying the MapReduce technique for two scientific data analyses: (i) High Energy Physics data analyses; (ii) Kmeans clustering and present CGL-MapReduce, a streaming-based MapReduce implementation and compare its performance with Hadoop.	Advantages - As the amount of data and the amount of computation increases, the overhead induced by a particular runtime diminishes. Even tightly coupled applications can benefit from the MapReduce technique if the appropriate size of data and an efficient runtime are used. In our future works, we are planning to improve the CGL-MapReduce in two key areas: the fault tolerance support;
15)	Partition based clustering of large datasets using MapReduce framework: An analysis of recent themes and directions	2018 FCI Journal	Tanvir Habib Sardar, Zahid Ansari	-It provides a comprehensive review of Hadoop and MapReduce and their components. -This paper aims to survey recent research works on partition based clustering algorithms which use MapReduce as their programming paradigm.	Clustering of large datasets using MapReduce greatly reduces the computation time. Does not have much effect for small dataset. Future scope: -converting traditional partial based clustering systems to MapReduce. design clustering algorithm is such a way that it can tune the fault

					tolerance of the system.

4.PROBLEM FORMULATION

The problems can be stated as in points as shown below:

- In most of the papers, number of clusters and already known a-priori. None of the papers propose any technique to find the optimum number of clusters for document clustering.
- The datasets in papers used have very structured documents. None of the paper propose any method to perform clustering on raw data or pdfs.
- None of the works have used to read the large number of documents in parallel using distributed environment.

5.RESULT AND DISCUSSION

Fig:1 Comparing Serial execution with Distributed execution we found a significant improvement 3 in time which was required to do the task. The graph gives the incite about the time taken when executing the program in different Parallel machines to the time taken by the program on executing it in serial manner. First graph is plotted between Data Size and the time take

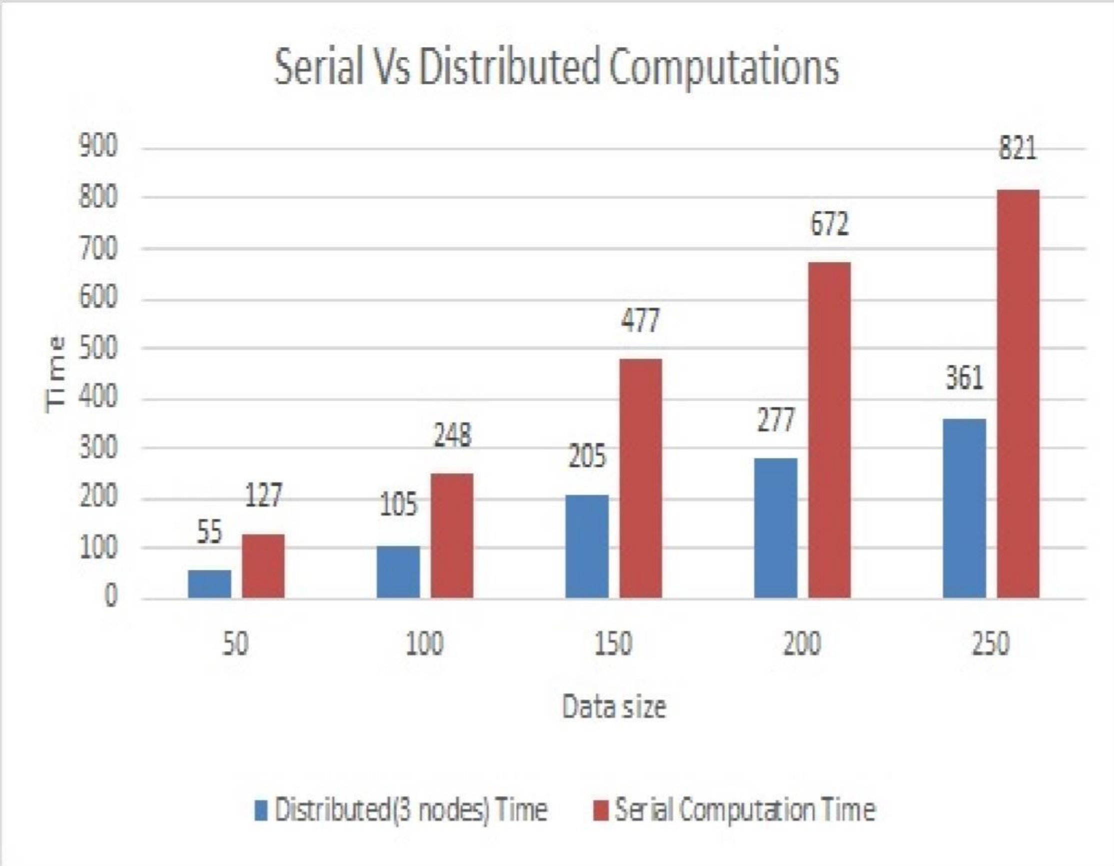

Figure 2: Serial vs Distributed Timings(in seconds)

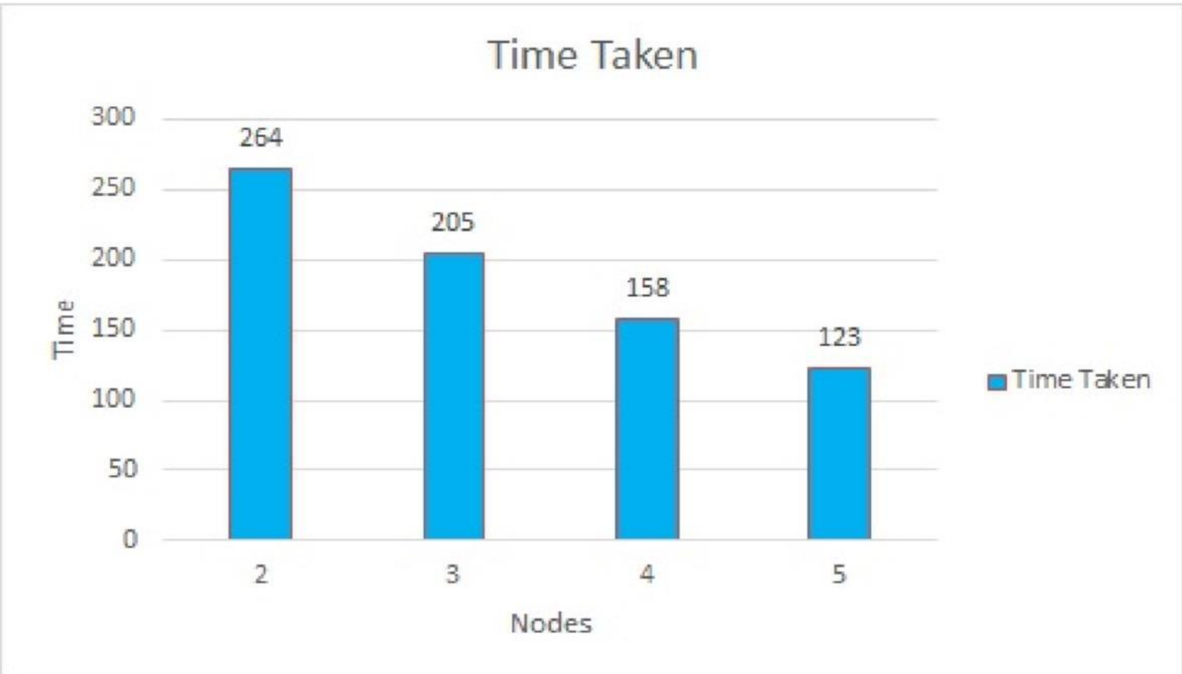

Figure 3: The Graphs shows the time taken by the different clusters.

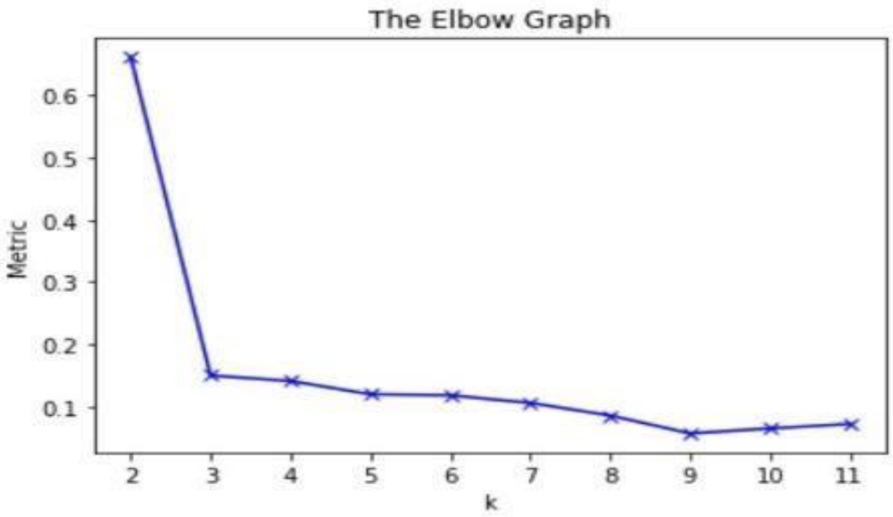

Figure 4: Elbow graph to find the optimum number of clusters;

So, overall we have done is: Initially data is extracted from the PDF files. Then based of the data of files the normalized term frequency matrix is constructed on which clustering will be performed. The proposed solution will be implemented in Hadoop where clustering for different values of 'k' will be performed on separate machines. All the instance will return the metric value calculated for respective value of 'k' to the master machine. The master machine will then plot the graph in the metric values received vs 'k'. Hence finally the optimum value of 'k' is chosen and the clustering is done one last time that the select value of 'k'.

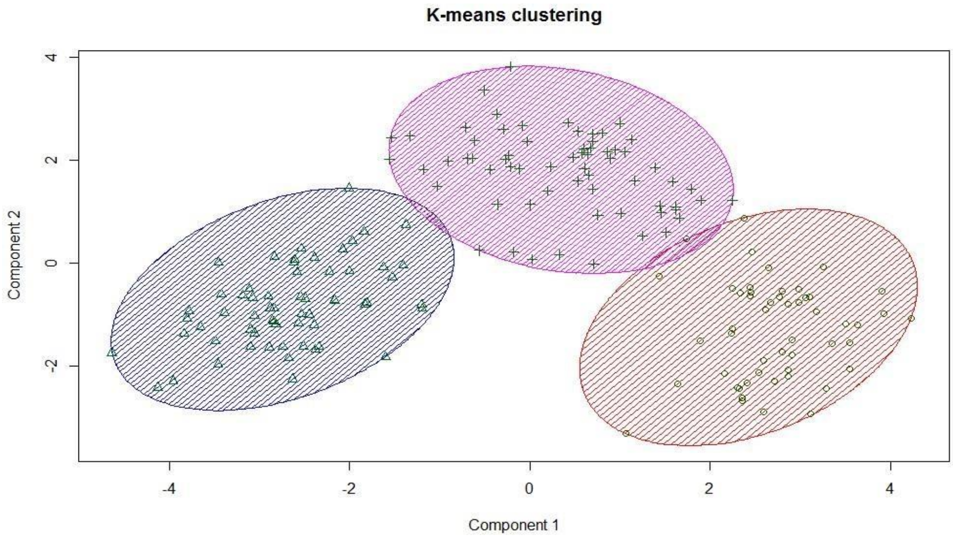

Figure 5: Final Result Of Clusters

6. CONCLUSIONS AND FUTURE WORK

Document clustering is a well-known and popular technique, but, due to lack of practicality it is not widely used. The proposed technique in this paper makes document clustering robust and overall ready to be used in any case. Many large corporates hire employees just to manage and sequence their data. Hence, such a tool that can do it efficiently and effectively has huge scope.

It can now be applied to unstructured set of pdf files which has increased the field of application of document clustering and the proposed technique. But at the same time, it is also important to factor in the other parameters that affect the performance of the model. Hence, a significance amount of further study is required to make the model robust and more efficient than now.

Future Prospects:

- Pre-processing of the documents can also be done parallelly. Tokenization, stop-word removal can be done when the pdf is converted to text.
- Custom metric can be created to suit the context of the dataset. Modified metrics can also be used and applied into a different study.
- More advanced clustering algorithms can be used like k-medoid and fuzzy clustering.
- Clustering itself can be implemented in distributed environment for better performance using Map-Reduce or in Spark.
- Nature of the ‘k’ vs metric graph can further be studied as it sometimes moves upwards as ‘k’ increases.
- Elbow point can be selected automatically while in current model it is picked manually. These results can be used as reference for the future study and advancement of this technique. The performance can be compared and the models can be tweaked for better efficiency and outcomes.

7 .REFERENCES

[1] Marcel R Ackermann, Marcus M¨artens, Christoph Raupach, Kamil Swierkot, Christiane Lammersen, and Christian Sohler. Streamkm++: A clustering algorithm for data streams. *Journal of Experimental Algorithmics (JEA)*, 17:2–4, 2012.

[2] Agrawal and Madhura Phatak. A novel algorithm for automatic document clustering. In *2013 3rd IEEE International Advance Computing Conference (IACC)*, pages 877–882. IEEE, 2013.

[3] Hameeza Ahmed, Muhammad Ali Ismail, and Muhammad Faraz Hyder. Performance optimization of hadoop cluster using linux services. In *17th IEEE International Multi Topic Conference 2014*, pages 167–172. IEEE, 2014.

[4] Prajesh P Anchalia, Anjan K Koundinya, and NK Srinath. Mapreduce design of k-means clustering algorithm. In *2013 International Conference on Information Science and Applications (ICISA)*, pages 1–5. IEEE, 2013.

[5] Chi-Ou Chen, Ye-Qi Zhuo, Chao-Chun Yeh, Che-Min Lin, and ShihWei Liao. Machine learning-based configuration parameter tuning on hadoop system. In *2015 IEEE International Congress on Big Data*, pages 386–392. IEEE, 2015.

[6] Jaliya Ekanayake, Shrideep Pallickara, and Geoffrey Fox. Mapreduce for data intensive scientific analyses. In *2008 IEEE Fourth International Conference on eScience*, pages 277–284. IEEE, 2008.

[7] Rui Maximo Esteves, Rui Pais, and Chunming Rong. K-means clustering in the cloud–a mahout test. In *2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications*, pages 514–519. IEEE, 2011.

[8] Tal Garfinkel, Ben Pfaff, Jim Chow, Mendel Rosenblum, and Dan Boneh. Terra: A virtual machine-based platform for trusted computing. In *ACM SIGOPS Operating Systems Review*, volume 37, pages 193–206. ACM, 2003.

[9] Dweepna Garg and Khushboo Trivedi. Fuzzy k-mean clustering in mapreduce on cloud based hadoop. In *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*, pages 1607–1610. IEEE, 2014.

[10] Kannan Govindarajan, Thamarai Selvi Somasundaram, Vivekanandan S Kumar, et al. Continuous clustering in big data learning analytics. In *2013 IEEE Fifth International Conference on Technology for Education (t4e 2013)*, pages 61–64. IEEE, 2013.

[11] Khaled M Hammouda and Mohamed S Kamel. Efficient phrase-based document indexing for web document clustering. *IEEE Transactions on knowledge and data engineering*, 16(10):1279–1296, 2004.

[12] Muhammad Faraz Hyder, Muhammad Ali Ismail, and Hameeza Ahmed. Performance comparison of hadoop clusters configured on virtual machines and as a cloud service. In *2014 International Conference on Emerging Technologies (ICET)*, pages 60–64. IEEE, 2014.

[13] Masakuni Ishii, Jungkyu Han, and Hiroyuki Makino. Design and performance evaluation for hadoop clusters on virtualized environment. In *The International Conference on Information Networking 2013 (ICOIN)*, pages 244–249. IEEE, 2013.

[14] Karthik Kambatla, Abhinav Pathak, and Himabindu Pucha. Towards optimizing hadoop provisioning in the cloud. *HotCloud*, 9(12):28–30, 2009.

[15 Jongyeop Kim, TK Ashwin Kumar, KM George, and Nohpill Park. Performance evaluation and tuning for mapreduce computing in hadoop distributed file system. In *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, pages 62–68. IEEE, 2015.

[16] Olga Kurasova, Virginijus Marcinkevicius, Viktor Medvedev, Aurimas Rapecka, and Pavel Stefanovic. Strategies for big data clustering. In *2014 IEEE 26th international conference on tools with artificial intelligence*, pages 740–747. IEEE, 2014.

[17] Rupesh Kumar Mishra, Kanika Saini, and Sakshi Bagri. Text document clustering on the basis of inter passage approach by using k-means. In *International Conference on Computing, Communication & Automation*, pages 110–113. IEEE, 2015.

[18] Sun Park, Dong Un An, and Choi Im Cheon. Document clustering method using weighted semantic features and cluster similarity. In *2010 Third IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning*, pages 185–187. IEEE, 2010.

[19] Aditya B Patel, Manashvi Birla, and Ushma Nair. Addressing big data problem using hadoop and map reduce. In *2012 Nirma University International Conference on Engineering (NUiCONE)*, pages 1–5. IEEE, 2012.

[20] Neepa Shah and Sunita Mahajan. Scalability analysis of semantics based distributed document clustering algorithms. In *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, pages 763–768. IEEE, 2017.

[21] Mihai Surdeanu, Jordi Turmo, and Alicia Ageno. A hybrid unsupervised approach for document clustering. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 685–690. ACM, 2005.

[22] Thaung Thaung Win and Lin Mon. Document clustering by fuzzy cmean algorithm. In *2010 2nd International Conference on Advanced Computer Control*, volume 1, pages 239–242. IEEE, 2010.

[23] Guanghui Xu, Feng Xu, and Hongxu Ma. Deploying and researching hadoop in virtual machines. In *2012 IEEE International Conference on Automation and Logistics*, pages 395–399. IEEE, 2012.

[24] Sen Xu, Zhimao Lu, and Guochang Gu. A fast spectral method to solve document cluster ensemble problem. In *2008 International Multisymposiums on Computer and Computational Sciences*, pages 180–183. IEEE, 2008.

[25] Yang Yang, Xiang Long, Xiaoqiang Dou, and Chengjian Wen. Impacts of virtualization technologies on hadoop. In *2013 Third International Conference on Intelligent System Design and Engineering Applications*, pages 846–849. IEEE, 2013.

[26] Kejiang Ye, Xiaohong Jiang, Yanzhang He, Xiang Li, Haiming Yan, and Peng Huang. vhadoop: A scalable hadoop virtual cluster platform for mapreduce-based parallel machine learning with performance consideration. In *2012 IEEE International Conference on Cluster Computing Workshops*, pages 152–160. IEEE, 2012.

[27] Xia Yue, Wang Man, Jun Yue, and Guangcao Liu. Parallel kmedoids++ spatial clustering algorithm based on mapreduce. *arXiv preprint arXiv:1608.06861*, 2016.

[28] Ming Zhao and Renato J Figueiredo. Experimental study of virtual machine migration in support of reservation of cluster resources. In *Proceedings of the 2nd international workshop on Virtualization technology in distributed computing*, page 5. ACM, 2007.

[29] Weizhong Zhao, Huifang Ma, and Qing He. Parallel k-means clustering based on mapreduce. In *IEEE International Conference on Cloud Computing*, pages 674–679. Springer, 2009.

Appendix:

Sample Code: The below sample code is used to run the Spark in the Python shell and it distributes the work to slaves and finally returns all work to Master.

```
#!/usr/bin/env python

# coding: utf-8

# In[1]:

import findspark

findspark.init()

import pyspark

from pyspark import SparkConf, SparkContext


conf = SparkConf().setMaster("local").setAppName("Pdf File")

sc = SparkContext(conf = conf)


# In[2]:


import io

from pdfminer.pdfinterp import PDFResourceManager, PDFPageInterpreter

from pdfminer.converter import TextConverter

from pdfminer.layout import LAParams

from pdfminer.pdfpage import PDFPage


 In[3]:


import time

import re
```

```
import math
import glob
# In[4]:
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import numpy as np
from scipy.spatial.distance import cdist
from sklearn.metrics import silhouette_score



# In[5]


path = 'D:\PDFfileall'
global pathlen
pathlen=len(path)
files = [f for f in glob.glob(path + "**/*.pdf")]
#print(files)
# In[6]:


global names
names=[]
global co
co=0
# In[7]:
def convert_pdf_to_txt(path):
    global co
    print(co)
    co=co+1
    rsrcmgr = PDFResourceManager()
    retstr = io.StringIO()
```

```
    codec = 'utf-8'
    laparams = LAParams()
    device = TextConverter(rsrcmgr, retstr, codec=codec, laparams=laparams)
    fp = open(path, 'rb')
    interpreter = PDFPageInterpreter(rsrcmgr, device)
    password = ""
    maxpages = 0
    caching = True
    pagenos = set()

    for page in PDFPage.get_pages(fp, pagenos, maxpages=maxpages,
                                  password=password,
                                  caching=caching,
                                  check_extractable=True):
        interpreter.process_page(page)

    text = retstr.getvalue(
    fp.close()
    device.close()
    retstr.close()
    text = re.findall('[a-zA-Z][a-zA-Z]+',text)
    #print(text)
    return text



# In[8]:



def read_files_from(file_list):
    termsDoc=[]
    for i in file_list:
```

```
        x=convert_pdf_to_txt(i)
        termsDoc.append([a.lower() for a in x])
    return termsDoc

def spark(files):
    n_parts = 2
    rdd1 = sc.parallelize(files, n_parts ) #distribute files among nodes
    ts=time.clock()
    list_of_pdf_strings = rdd1.mapPartitions(read_files_from).collect()
    ts=time.clock()-ts
    print(ts)
    return list_of_pdf_strings
# In[9]:
"""type(list_of_pdf_strings)
for i in files:
    names.append(i[pathlen+1:])""
# In[10]:
def FilterDoc(files):
    termsDoc=[]
    names=[]
    for i in files:
        x=convert_pdf_to_txt(i)
        #x = re.findall('[a-zA-Z]{2}[a-zA-z]*',x)
        termsDoc.append([a.lower() for a in x])
        names.append(i[pathlen+1:])
    return termsDoc, names
# In[11]:
global sw
sw=["i", "me", "my", "myself", "we", "our", "ours", "ourselves", "you", "your", "yours", "yourself",
"yourselves", "he", "him", "his", "himself", "she", "her", "hers", "herself", "it", "its", "itself", "they",
"them", "their", "theirs", "themselves", "what", "which", "who", "whom", "this", "that", "these",
"those", "am", "is", "are", "was", "were", "be", "been", "being", "have", "has", "had", "having", "do",
```

```
"does", "did", "doing", "a", "an", "the", "and", "but", "if", "or", "because", "as", "until", "while", "of",
"at", "by", "for", "with", "about", "against", "between", "into", "through", "during", "before", "after",
"above", "below", "to", "from", "up", "down", "in", "out", "on", "off", "over", "under", "again",
"further", "then", "once", "here", "there", "when", "where", "why", "how", "all", "any", "both",
"each", "few", "more", "most", "other", "some", "such", "no", "nor", "not", "only", "own", "same",
"so", "than", "too", "very", "s", "t", "can", "will", "just", "don", "should", "now"]

tdoc=[]

termsDoc=[]

completeList=[]

# In[12]:

termsDoc,names=FilterDoc(files)

print("\n\nThe Pdfs available are:\n")

s=0

for i in names:

  print(s+1,") ",i)

  s=s+1

# In[13]:

def RemStopWords(termsDoc):

  global sw

  termsDoc1=termsDoc

  for i in range(len(names)):

    termsDoc[i]=[a for a in termsDoc[i] if a not in sw]

  return termsDoc,termsDoc1

# In[14]

termsDoc,termsDoc1=RemStopWords(termsDoc)

def CreatingList(termsDoc):

  global sw

  global completeList

  trial=[]

  #trial=ai+wm+dm

  #print(len(trial),len(ai))

  trial = [a for a in termsDoc]

  for i in trial:
```

```
        for j in i:
            if j not in sw:
                completeList.append(j)
    completeList=list(set(completeList))
CreatingList(termsDoc)
# In[15]:
def BooleanMatrix(termsDoc1):
    global completeList
    lenOfDocs=[]
    for i in termsDoc1:
        lenOfDocs.append(len(i))
    l=-1
    bools=[]
    for k in termsDoc:
        temp=[]
        l=l+1
        for i in completeList:
            if(i in k):
                x=k.count(i)
                temp.append(x/lenOfDocs[l])
            else:
                temp.append(0)
        bools.append(temp)
    mat=[completeList]
    for i in bools:
        mat.append(i)
    return mat
# In[16]:
mat=BooleanMatrix(termsDoc1)
def idfVector(mat):
    global completeList
```

```
    idf=[]
    for i in range(len(completeList)):
        c=0
        for j in range(len(names)):
            #print(mat[j+1][i])
            if(mat[j+1][i]>0):
                c=c+1
        if(c!=0):
            idfx=math.log((1+3)/c)
        else:
            idfx=0
        idf.append(idfx)
    return idf
# In[17]:
print(mat)
# In[18]:
words=mat[0]
mat=mat[1:]
M=pd.DataFrame(mat,columns=words)
#print(M)
X = M.iloc[:,:].values
#print(X)
# kmeans = KMeans(n_clusters =4,init = 'k-means++',max_iter=100000,n_init=10)
# y_kmeans = kmeans.fit_predict(X)
# print(y_kmeans)
# pd.DataFrame(names,y_kmeans)
# for i in names:
#     print(i)


# In[19]:
X=pd.DataFrame(mat)
```

```
# In[20]:
print(len(X.columns))
# In[21]:
import pickle
# In[22]:
def save_training_data(X):
   pickle_out=open("x.pickle","wb")
   pickle.dump(X,pickle_out)
   pickle_out.close()
save_training_data(X)
# In[23]:
def load_data():
   pickle_in=open("x.pickle","rb")
   X=pickle.load(pickle_in)
   return X
X=load_data()
# In[24]:
global k
k=[]
arr="2,3,4,5,6,7,8,9,10,11".split(',')
# In[25]:
print(arr)
# In[26]:
def multiple_kmeans(arr):
   termsDoc=[]
   global distortions
   distortions=[]
   for i in arr:
      global k
      #x=convert_pdf_to_txt(i)
```

```
        #termsDoc.append([a.lower() for a in x])
        kmeans = KMeans(n_clusters =int(i),init = 'k-means++',max_iter=10000,n_init=10)
        kmeans.fit(X)
        y_kmeans = kmeans.fit_predict(X)
        #k.append(y_kmeans)
        #distortions.append(sum(np.min(cdist(X, kmeans.cluster_centers_, 'euclidean'), axis=1)) /
X.shape[0])
        #distortions.append(kmeans.inertia_)
        label = kmeans.labels_
        distortions.append(silhouette_score(X, label, metric='euclidean'))
    return distortion
n_parts = 3
rdd = sc.parallelize(arr, n_parts ) #distribute files among nodes
ts=time.clock()
Y = rdd.mapPartitions(multiple_kmeans).collect()
ts=time.clock()-ts
print(ts)
# In[27]:
print(Y)
# In[28]:
plt.plot(arr, Y, 'bx-')
plt.xlabel('k')
plt.ylabel('Metric')
plt.title('The Elbow Graph')
plt.show()
# In[ ]:
K=int(input("Enter the value of K: "))
kmeans = KMeans(n_clusters =K
                ,init = 'k-means++',max_iter=100000,n_init=10)
y_kmeans = kmeans.fit_predict(X)
# In[ ]
```

```
print(y_kmeans)

# In[ ]

Result=pd.DataFrame(names,y_kmeans)
```