Volume 4, Issue 11, November 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Distributed Document Clustering Using K-Means

Neepa Shah
Assistant Professor, IT Department, DJSCE,
MU, Mumbai, India

Dr. Sunita Mahajan
Principal, Institute of Computer Science,
M.E.T., Mumbai, India

Abstract- Document clustering, one of the traditional data mining techniques, is an unsupervised learning paradigm where clustering methods try to identify inherent grouping of the text documents.The importance of document clustering emerges from the massive volumes of textual documents created. Also, with more and more development of information technology, data set in many domains is reaching beyond peta-scale; making it difficult to work with the document clustering algorithms in central site and leading to the need of increasing the computational requirements. The concept of distributed computing thus; is explored for document clustering giving rise to distributed document clustering. Here, we propose distributed document clustering usingHadoop and MapReduce. We implemented K-means and tested on single node and then modified the map, reduce functions to run over cluster of three machines. We tested on two datasets consisting of 20000 documents (20-NewsGroups) and 21578 documents (Reuters-21578). The results show that timing requirement for clustering reduces with addition of nodes in the cluster.

Keywords: Document Clustering, K-means document clustering, Hadoop,MapReduce, Distributed Document Clustering

I. INTRODUCTION

The technology advancement in computer hardware technology, powerful computers, storage media, and data collection equipments has provided a tremendous growth in the volume of the text documents. With the increase in the number of electronic documents, it is hard to organize, analyze and present these documents efficiently by putting manual effort [1]. These have brought challenges for the effective and efficient organization of text documents automatically [2]. For this, document clustering, an unsupervised machine learning approach is used.

Document clustering organizes documents into different groups called as clusters, where the documents in each cluster share some common properties according to defined similarity measure. The fast and high-quality document clustering algorithms play an important role in helping users to effectively navigate, summarize, and organize the information [3].

There are various challenges and requirements of clustering algorithm to increase the quality of clusters. These include finding a suitable model to represent the document, reducing the high dimension of text documents, allowing overlapping of document clusters, associating a meaningful label to each final cluster, estimating the number of clusters, improving the scalability, and extracting semantics from text.

Here, we propose the model for distributed document clustering to address scalability issue. For this, we are using open source framework for distributed computing namely Hadoop and MapReduce.

The paper is organized as: Section II highlights related work in the area document clustering and distributed document clustering. Section III gives single node and multi node Hadoop installation. Section IV deals with K-means and distributed K-means over single node and multi node Hadoop with results on two datasets as mentioned in section V. We conclude the paper with mention of future work in Section VI.

II. RELATED WORK

In this section, related work in the area of document clustering like various basic algorithms and methods, parallel and distributed document clustering techniques, and Hadoop / MapReduce framework for document clustering is briefly discussed.

Various algorithms are proposed for document clustering starting from Croft; which clustered only titles. Then k-means (KM), hierarchical agglomerative clustering (HAC) and variations of KM and HAC algorithms are proposed over years. Concepts of frequent-itemset, fuzzy theory, neural network, genetic algorithm, self-organizing map, non-negative matrix factorization etc. are also studied for increasing efficiency of document clustering. These algorithms tried to address the issues like reducing dimensionality, suggesting better model for document representation, estimating number of clusters etc. These algorithms are studied and compared in [4].

The datasets are increasing at exponential order. So only these algorithmic and conceptual changes are not enough in current scenario of large datasets. This is because huge volume of documents produced due to advancement in technology possesses two key characteristics: (a) data is no longer maintained in central databases and (b) computers processing these data are no longer central supercomputers, but rather a huge network of computers. In this scenario, centralized mining cannot scale to the magnitude of the data [5].

Storage and computing of mass documents in a distributed system is an alternative approach. Scalable and easy-to-use tools for distributed processing are also emerging over a period of time. The most popular one of these emerging technologies is Hadoop-open source software framework for data intensive distributed applications.

Hadoop is a fault-tolerant distributed system for data storage which is highly scalable. This scalability and reliability is due to its unique file system; Hadoop Distributed File System (HDFS). MapReduce is parallel programming abstraction and runtime support for scalable data processing on Hadoop. Thus, Hadoop provides a reliable shared storage and analysis system. The storage is provided by HDFS and analysis by MapReduce[6]. Various parallel and distributed algorithms and role of Hadoop and MapReduce in distributed clustering applications is covered in detail in [7].

III. HADOOP: SINGLE-NODE AND MULTI-NODE

Hadoop is sub-project of Lucene (a collection of industrial-strength search tools), under the umbrella of the Apache Software Foundation. It is an open source software framework to support data intensive distributed applications. It parallelizes data processing across many nodes (computers) in a compute cluster, speeding up large computations [8].

Hadoop Core includes:

- Hadoop Distributed File System (HDFS) -> distributes data: It is a distributed file system designed to hold very large amounts of data (terabytes or even petabytes), and provide high-throughput access to this information. Here, Files are stored in a redundant fashion across multiple machines to ensure their durability to failure and high availability to very parallel applications[9].
- Map/Reduce -> distributes application: It is a parallel and distributed solution approach developed by Google for processing large datasets. It is a software framework for processing large data sets in a distributed fashion over a several machines. It is utilized by Google and Yahoo to power their web search[10].

MapReduce is a software framework for processing large data sets in a distributed fashion over several machines. The core idea behind MapReduce is mapping data set into a collection of <key, value> pairs, and then reducing all pairs with the same key. While writing code there are 2 scripts: the map script, and the reduce script. The framework splits the input into parts, passing each part to a different machine. Each machine runs the map script on the portion of data assigned to it. The map script maps this data to <key, value> pairs according to the specifications. These generated <key, value> pairs are shuffled i.e. pairs with the same key are grouped and passed to a single machine, which will run the reduce script over them [10].Thus, Map transforms a set of data into key-value pairs and Reduce aggregates this data into a scalar.

For our experiments, we first set up a pseudo-distributed, single-nodeHadoop[12] and then distributed, multi-nodecluster of three nodes running on Ubuntu Linux [13].

We then ran WordCount on both the configurations. WordCount example reads text files and counts how often words occur. The input is text files and the output is text files, each line of which contains a word and the count of how often it occured, separated by a tab. Here, each mapper takes a line as input and breaks it into words. It then emits a key/value pair of the word. Each reducer sums the counts for each word and emits a single key/value with the word and sum. As an optimization, the reducer is also used as a combiner on the map outputs. This reduces the amount of data sent across the network by combining each word into a single record[14].

We have tested WordCount example using two text sizes: 5.93 MB and 644 MB. For single node setup, three ebooks from Gutenberg Project [15]are taken and ran.For multimode seven eBooks are considered. The original text sie is 5.93 MB. We modified this to increase the size of input to get 644 MB.Below we give analysis of both these inputs

Table Iand Fig.1.

Table I: wordcount Program Execution Time on Single node and Multi node Hadoop

Text Size	1 Node	2 Nodes	3 Nodes
5.93 MB	49	32	28
644 MB	212	108	84

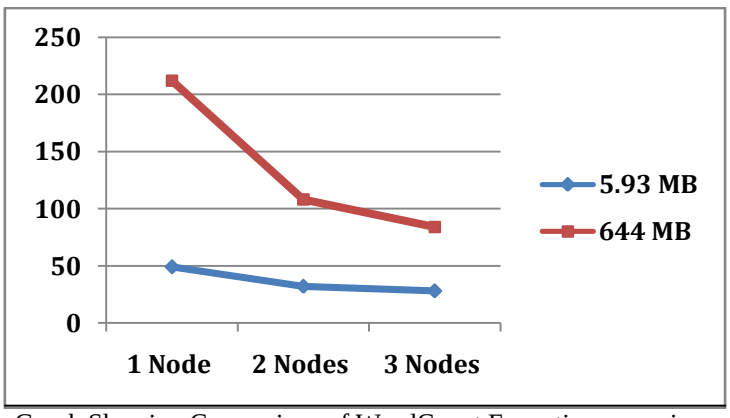

Fig.1: Graph Showing Comparison of WordCount Execution on various nodes

IV. DISTRIBUTED K-MEANS OVER HADOOP

In this section, we describe K-means document clustering. Then we will cover distributed K-means using Hadoop framework.

A. Direct K-Means Clustering

K-means document clustering comes under partitional technique of clustering where one-level (un-nested) partitioning of the data points is created. If K is the desired number of clusters, then partitional approaches typically find all K clusters at once. K-means is based on the idea that a center point can represent a cluster. In particular, for K-means we use the notion of a centroid, which is the mean or median point of a group of points [16]. Basic k-means algorithm is given below[17].

Input: K: number of cluster, D: Top N documents
Output: K clusters of documents
Algorithm:

Generate K centroids C1, C2, ..,Ck by randomly choosing K documents from D
Repeat until there is no change in cluster between two consecutive iterations
for each document di in D
for j = 1 to K
Sim(Cj,di) = Find cosine similarity between di and Cj
end for
Assign di to cluster j for which Sim(Cj,di) is maximum
end for
Update centroid for each cluster
end loop
end K-Means

B. K-means and Hadoop/ Distributed K-Means

We show here distributed version of K-means algorithm[18]. The process of clustering starts with vectorization; wherein we do pre-processing of text corpus like stemming, stop-word removal etc. It produces vector which is "termsXdocuments" matrix i.e. tfidf values. We have tested on 20_newsgroups text corpus[19]; which contains 20 subdirectories each with 1000 documents. So, vector file has 20 X 1000 lines i.e. for 20000 documents. This process is illustrated in Fig.2.

```
hduser@neepa-Latitude-D630:~/kmeans_dc$ java -jar ProcessCorpus.jar
Enter the directory where the corpus is located: 20_newsgroups
Enter the name of the file to write the result to: v1
Enter the max number of docs to use in each subdirectory: 100
20_newsgroups
Counting the number of occurs of each word in the corpus...Found 45397 unique words in the corpus.
How many of these words do you want to use to process the docs? 10000
Firefox Web Browser
ictionary.
Converting the corpus to a list of vectors...Done vectorizing all of the docs!
```

Fig.2: Document Dataset (20_newsgroups) Preprocessing to Generate Vectors

This vector is used to calculate initial set of centroids from the data. This is done by giving vector and number of clusters as input to produce cluster centroids. This initial set of cluster centroids is simply randomly sampled from the 'vectors' file. This process is shown inFig.3.

```
hduser@neepa-Latitude-D630:~/kmeans_dc$ java -jar GetCentroids.jar
Enter the data file to select the clusters from: v1
Enter the name of the file to write the result to: c1
Enter the number of clusters to select: 20
..Done selecting centroids.
```

Fig.3: Getting Initial Centroids from Vectors (20_newsgroups Dataset)

To run k-means over hadoop, we need to move these files 'vectors' and 'cluster centroids' into HDFS file system. This is done using mkdir and copyFromLocal command on Hadoop as shown in Fig. 4.

```
hduser@neepa-Latitude-D630:~/kmeans_dc$ hadoop dfs -mkdir /d1
Warning: $HADOOP_HOME is deprecated.

hduser@neepa-Latitude-D630:~/kmeans_dc$ hadoop dfs -mkdir /c1
Warning: $HADOOP_HOME is deprecated.

hduser@neepa-Latitude-D630:~/kmeans_dc$ hadoop dfs -copyFromLocal v1 /d1
Warning: $HADOOP_HOME is deprecated.

hduser@neepa-Latitude-D630:~/kmeans_dc$ hadoop dfs -copyFromLocal c1 /c1
Warning: $HADOOP_HOME is deprecated.
```

Fig. 4: Copying Local Data to HDFS

As seen in earlier, MapReduce programming framework needs a mapper program and reducer program. These files are bundled in MapRedKMeans.jar. So, next, is to run MapReduce program using command "hadoop jar MapRedKMeans.jar KMeans /data /clusters 1". Here, /data and /clusters are HDFS directories storing vectors and initial

clusters. '1' indicates number of iteration. If we give last parameter as '3', it will run 3 iterations of the KMeans algorithm. This means that three separate MapReduce jobs will be run in sequence. The centroids produced at the end of iteration 1 will be put into the HDFS directory "/clusters1", those from the end of iteration 2 in "/clusters2", and those from the end of iteration 3 in "/clusters3". When it completes, the results can be examined by running: "java -jar GetDistribution.jar". This will count how many of each of the 20 types of newsgroups was put into each cluster. It is given below in Fig.5 which shows 20 clusters numbered as cluster 0 to cluster 19.

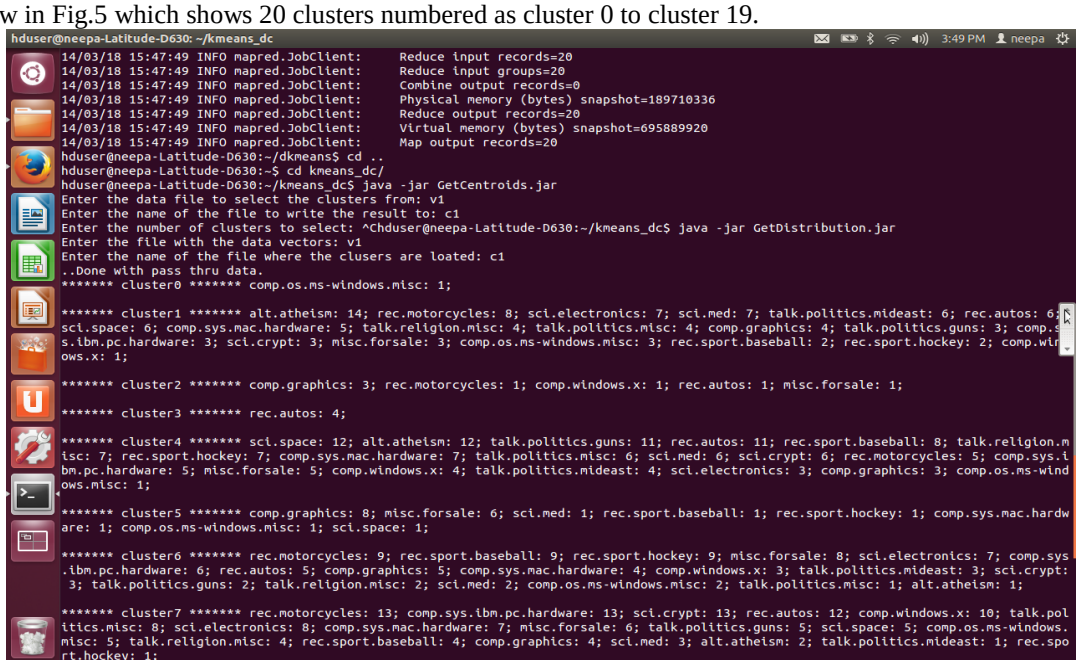

Fig.5: Distribution of Documents in Various Clusters (20_newsgroups Dataset)

V. EXPERIMENTS AND RESULTS

We ran one iteration of distributed version of K-Means on 20_newsgroups dataset by gradually increasing number of documents from 2000 to 20000. We ran on single-node Hadoop and then on 2-nodes and 3-nodes clusters. Result of time taken (in seconds) for this set-up is given below in Table II. Further, we plot line graph to analyze performance improvement while increasing number of nodes. This graph is depicted in Fig.6 . Here, y-axis indicates time in seconds and x-axis shows 10 records for 2000 to 20000 documents by increment of 2000 each. Figure indicates that performance is enhanced using 3-nodes compared to 1 node.

Table II: Time taken in seconds to process 20_newsgroups dataset

Number of Docs	1 Node	2 Nodes	3 Nodes
2000	24	18	19
4000	30	24	25
6000	35	25	26
8000	43	25	26
10000	51	32	27
12000	51	33	28
14000	60	35	34
16000	72	39	35
18000	86	40	36
20000	89	46	37

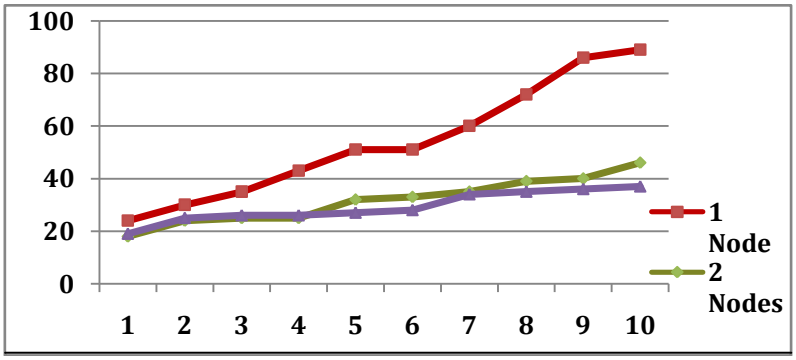

Fig.6: Performance Improvement by increasing Number of Nodes (20_newsgroups)

After this testing, we also tried dataset Reuters-21578[20]. This dataset is available as 22 SGML files, with 21 files containing description of 1000 documents and 1 file with description of remaining 578 documents. We used Mahout [21] to convert SGML files into text files. The process and result is shown below inFig.7.

```
hduser@neepa-Latitude-D630:~/mahout-work$ cd reuters-sgm/
hduser@neepa-Latitude-D630:~/mahout-work/reuters-sgm$ ls
all-exchanges-strings.lc.txt  feldman-cia-worldfactbook-data.txt  reut2-003.sgm  reut2-009.sgm  reut2-015.sgm  reu
all-orgs-strings.lc.txt       lewis.dtd                          reut2-004.sgm  reut2-010.sgm  reut2-016.sgm
all-people-strings.lc.txt     README.txt                         reut2-005.sgm  reut2-011.sgm  reut2-017.sgm
all-places-strings.lc.txt     reut2-000.sgm                      reut2-006.sgm  reut2-012.sgm  reut2-018.sgm
all-topics-strings.lc.txt     reut2-001.sgm                      reut2-007.sgm  reut2-013.sgm  reut2-019.sgm
cat-descriptions_120396.txt   reut2-002.sgm                      reut2-008.sgm  reut2-014.sgm  reut2-020.sgm
hduser@neepa-Latitude-D630:~/mahout-work/reuters-sgm$ cd ..
hduser@neepa-Latitude-D630:~/mahout-work$ cd ..
hduser@neepa-Latitude-D630:~$ bin/mahout org.apache.lucene.benchmark.utils.ExtractReuters mahout-work/reuters-sgm
out
bash: bin/mahout: No such file or directory
hduser@neepa-Latitude-D630:~$ mahout/mahout-distribution-0.6/bin/mahout org.apache.lucene.benchmark.utils.ExtractR
euters-sgm mahout-work/reuters-out
MAHOUT_LOCAL is not set; adding HADOOP_CONF_DIR to classpath.
Running on hadoop, using HADOOP_HOME=/usr/local/hadoop
No HADOOP_CONF_DIR set, using /usr/local/hadoop/conf
MAHOUT-JOB: /home/hduser/mahout/mahout-distribution-0.6/mahout-examples-0.6-job.jar
Warning: $HADOOP_HOME is deprecated.
```

Fig.7: Mahout Command to Convert SGML files to text files

Results of time taken (in seconds) to cluster this dataset again for 1, 2 and 3-nodes for 1 iteration is given below in Table III. We plotted line graph to analyse performance improvement while increasing number of nodes. This graph is depicted in Fig.8. Figure indicates that performance is enhanced using 3-nodes compared to 1 node.

Table III: Time taken in seconds to process Reuters-21578 dataset

Number of Docs	1 Node	2 Nodes	3 Nodes
2000	20	16	17
4000	21	17	17
6000	22	17	18
8000	30	22	20
10000	32	24	22
12000	34	25	21
14000	34	26	22
16000	35	27	22
18000	35	27	22
20000	48	29	23
22000	56	32	24

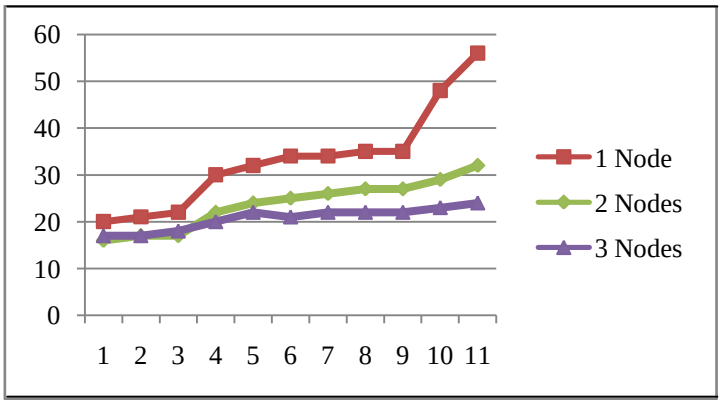

Fig.8: Performance Improvement by increasing Number of Node (Reuters-21578)

VI. CONCLUSION AND FUTURE WORK

As the volume of information continues to increase, there is growing interest in helping people better find, filter and manage these resources. Text clustering, which is the process of grouping documents having similar properties based on semantic and statistical content, is an important component of document organization and management. But, clustering of documents according to semantic features is a challenging problem in text data mining. Due to rapid increase in digital copies of data, scalability is also an issue.

To handle scalability issue, distributed document clustering using Hadoop and MapReduce framework is studied and implemented. We ran Hadoop over three nodes and executed Wordcount over large text dataset to see the difference in performance. The K-Means document clustering is then tested on single node, followed by 3 nodes setup. For larger datasets, like reuters-21578 and 20-Newsgroups, it proved to optimize performance.

Bisecting K-Means is found to be better than K-means; so further we propose to implement Bisecting K-Means and compare the performance of K-Means and Bisecting K-Means using evaluation parameters like entropy and purity. We can also check adding more nodes and testing on still large datasets like RCV1.

Further, semantics can be added to increase the quality of document clustering. For this, we propose to use Stanford coreNLP, Lexicalized parser and POS tagger to add statistical and semantic tagging to the text. After this semantic similarity measures can be applied to cluster the documents.

REFERENCES

[1] RekhaBaghel and Dr. RenuDhir, "A Frequent Concepts Based Document Clustering Algorithm," *Int'l Journal of Computer Applications*, vol. 4, No.5, pp. 0975 – 8887, Jul. 2010

[2] Huang, "Similarity measures for text document clustering," *Proc. of the 6th New Zealand Computer Science Research Student Conference NZCSRSC*, pp. 49-56, 2008.

[3] Nicholas O. Andrews and Edward A. Fox, "Recent developments in document clustering," *Technical report published by citeseer*, pp. 1-25, Oct. 2007

[4] Neepa Shah and Sunita Mahajan, "Document Clustering: A Detailed Review," *Int'l Journal of Applied Information Systems*, Vol. 4, No. 5, pp.30-38, Oct. 2012

[5] K. Hammouda, "Distributed Document Clustering and Cluster Summarization in Peer-to-Peer Environments," *PhD Thesis*, Dept. of System Design Engineering, University of Waterloo, 2007

[6] Tom White, "Hadoop: The Definitive Guide," *O'Reilly Media*, Inc., 2009

[7] Neepa Shah and Sunita Mahajan, "Semantics Based Distributed Document Clustering: Proposal," *Int'l Journal of Computer Science Engineering and Information Technology Research*, Vol. 3, Issue 2, pp. 379-388, Jun. 2013

[8] Brandeis University, Networks and Distributed Computing, "Introduction to Hadoop," [online], available at http://www.cs.brandeis.edu/~rshaull/cs147a-fall-2008/hadoop-intro/, 2008

[9] Yahoo Developer Network, "Module 2: The Hadoop Distributed File System,", [online], available at http://developer.yahoo.com/hadoop/tutorial/module2.html

[10] Lars Vogel, Version 0.4, "MapReduce Introduction – Tutorial," [online], available at http://www.vogella.com/tutorials/MapReduce/article.html, Oct. 2010

[11] Diana MacLean, "A Very Brief Introduction to MapReduce," [online], available at http://hci.stanford.edu/courses/cs448g/a2/files/map_reduce_tutorial.pdf, 2011

[12] Michael G. Noll, Applied Research, Big Data, Distributed Systems, Open Source, "Running Hadoop on Ubuntu Linux (Single-Node Cluster)", [online], available at http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/

[13] Michael G. Noll, Applied Research, Big Data, Distributed Systems, Open Source, "Running Hadoop on Ubuntu Linux (Multi-Node Cluster)", [online], available at http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/

[14] WordCount example, [online], available at http://wiki.apache.org/hadoop/WordCount

[15] Project Gutenberg, [online], available at http://www.gutenberg.org/ebooks

[16] Michael Steinbach, George Karypis, and Vipin Kumar, "A comparison of document clustering techniques," *In KDD Workshop on Text Mining*, 2002

[17] Hemal Khatri, "Project C", CSE 494/598, [online] available at, http://rakaposhi.eas.asu.edu/cse494/f05-projects/ProjC_Hemal.pdf

[18] K-Means over Hadoop, [online], available at http://cmj4.web.rice.edu/MapRedKMeans.html

[19] Tom Mitchell, School of Computer Science, Carnegie Mellon University, [online], available at, https://archive.ics.uci.edu/ml/machine-learning-databases/20newsgroups-mld/

[20] Reuters-21578 Text Categorization Collection Data Set, UCI Machine Learning Laboratory, available at http://archive.ics.uci.edu/ml/datasets/Reuters-21578+Text +Categorization+Collection

[21] "What is Apache Mahout?", [online], available at http://mahout.apache.org/

