Analysis of Document Clustering Using K-means Algorithm with Cosine Similarity for Large Scale Text Documents With and Without Hadoop

Salina Dangol, Suresh Pokhrel
M.E. (Computer), M.E. (Computer)
Nepal College of Information Technology
salinadangol1@gmail.com
suresh@ncit.net.np

Abstract—With the advancement in technology there has been continuous increment in the datasets produced in unstructured format. The datasets are the documents consisting of the information in the different formats like text, portable document format (pdf), Extensible markup language (xml), audio, video etc. There is need of organization of such datasets into semantic group for the information retrieval in optimized way. Hadoop is popular open source software framework which support data intensive distributed applications and Map Reduce is parallel programming technique and runtime support for scalable data processing on Hadoop.

This paper is the analysis of performance of Hadoop for Document Clustering in the distributed system for large datasets. This is implemented using K-means clustering algorithm with cosine similarity for feature extraction.

Keywords—Distributed System, Document Clustering, Hadoop, Map Reduce, Cosine Similarity

I. INTRODUCTION

The data and information has always been the major source of the information retrieval in any field like education, scientific researches, business, medical research, management, economics, data management, human resource management, images processing and so on. Each day numerous data are generated and processed for the accomplishment of different activities throughout the world. This information is the vital from which intensive knowledge can be retrieved. The process of obtaining the knowledge from the mass of data and information is called as information retrieval.

Organizing the unstructured documents into the meaningful group is the sub problem of information retrieval in which there is need to learn about the general content of data [1].

Unstructured data are non-database data that includes text-based documents like word processing, presentations, emails, blogs, wikis, tweets, web pages, web components (read/write) web and video/audio files. Unstructured data are found in office productivity suites, Content management systems, digital asset management systems, web content management systems like wikis, blogs, comment, discussion boards and social networking tools like twitter, instant messengers, Facebook etc.

Data mining is the process of extracting the implicit, previously unknown and potentially useful information from data. Document clustering, subset of data clustering, is the technique of data mining which includes concepts from the fields of information retrieval, natural language processing, and machine learning. 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.

Hadoop is the most effective analytic tool used for processing large sets of data. It is designed to process, store and analyse petabytes and Exabyte of distributed, unstructured and structured data. Hadoop works in the nutshell. It breaks data into pieces and stored them into Hadoop distributed file system [HDFS] which can scale to hundreds of nodes on a single cluster and analyses and process millions of files in a single instance improving efficiency. It is cost effective and scalable method of storing, manipulating and querying data.

Data increasing trend is growing exponentially at an unprecedented space. These volume of data are of great essence with meaningful conclusions. These data are can be used to obtain essential information regarding geographical analysis, health status, education system, social causes and problems, cybercrime and internet threats, fraud and malicious intent and many more. In order to retrieve valuable information these data are needed to be processed.

Data are processed using various analytic tool and methods one of which includes text mining. Text mining is part of information retrieval and is used to retrieve information. In text mining the data are processed using the different techniques and one of them is clustering. Among clustering k means clustering is used with cosine similarity for feature extraction.
II. METHODOLOGY

This research will be focused on determining the feature extraction of the document on the basis of cosine similarity, implementation of vector space model (VSM) for pre-processing of the documents, applying the clustering algorithm and applying the Hadoop with MapReduce.

A. Methodology Adapted

i. The documents required are indexed using apache Lucene.
ii. Pre-processing is carried on, which includes the different methods to reduce the document size. It includes:
   a. Extraction of words.
   b. Removing stop words using Key Phrase Extraction Algorithm.
   c. Applying stemming using Porter Algorithm.
iii. Frequency matrix is generated using vector space model.
iv. Cosine similarity is applied for distance measure.
v. K-means clustering algorithm is applied.
vi. Then each job is submitted and processed using Hadoop platform.
vii. The performance measurement is done on k-means clustering.

B. Block Diagram

This phase consists of three further process. They are extracting the words, stop words removal and stemming. After pre-processing phase, the actual document is constructed. This document consists of the required unique terms after the removal of the stop words and stemming.

**Vector Space Model**

The vector space model (VSM) represents the derived terms as vectors providing the easy way for implantation for the desired work. Vector space model is an algebraic model for representing text documents as vectors of identifiers such as index term. It is used in information filtering, information retrieval, indexing and relevancy rankings [2]. This model focuses on the retrieval of the terms on the basis of the presence of each unique term present in given document.

In VSM, a collection of d documents described by t terms can be represented as a tex matrix ‘A’, commonly called term-document matrix. The column vectors are called document vector representing the documents in the collection and the row vectors are called term vectors representing the indexed terms from the documents.

**Table 1. Term-Document Matrix**

<table>
<thead>
<tr>
<th>Term</th>
<th>Document 1</th>
<th>Document 2</th>
<th>Document 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Term 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Term 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Term 3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Binary Weighting**

The term occurrence im VSM refers to binary weighting method where if \( a_{ij} = 1 \) means term i occurs in document j, \( a_{ij} = 0 \) meaning that the term ‘i’ doesn’t occur in document ‘j’. The binary weighting informs about the fact that a term is somehow related to a document but carries no information on the strength of the relationship.

**Term Frequency**

Term Frequency (tf) Weighting measures how frequently a term occurs in a document. It refers to the number of occurrences of the word in the document which represents the value of the vector dimension for the word. Thus the term frequency is often divided by the document length (i.e the total number of terms in the document) as a way of normalization. In this scheme \( a_{ij} = tf_{ij} \) where \( tf_{ij} \) denotes how many times term ‘i’ occurs in document ‘j’. This frequency for each term in the document is also called as local weight.

Mathematically,

\[
tf_{ij} = \frac{\text{number of times term 'i' appears in document 'j'} }{\text{total number of term 'i' s in document 'j'}}
\]  

(1)

**Inverse Document Frequency**

Inverse Document Frequency (idf) measures how important is the term is. While computing term frequency (tf), all terms are considered equally important. This scheme represents the scaling factor, or the importance of term ‘i’. The obtained value decreases as the number of document containing the term increases. The frequency obtained for each term is called global weight.
Mathematically, \( \text{idf}_i = \log \frac{\text{total number of documents}}{\text{number of documents with term } i} \) (2)

**Term Frequency- Inverse Document Frequency**

Term Frequency- Inverse Document Frequency (tfidf) weighting is the statistical measure used to evaluate how important a word is to the document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document. This scheme aims at balancing the local and the global term occurrences in the documents.

Mathematically,
\[
\text{tfidf} = \text{tf}_i \times \text{idf}_i \quad \text{………………….eqn(3)}
\]
where, \( \text{tf}_i \) and \( \text{idf}_i \) can be obtained from eqn(1) and eqn(2).

**Cosine Similarity**

Cosine Similarity is the common measure of similarity between two vectors which measures the cosine of the angle between them. In a \( (t \times d) \) term-document matrix \( A \), the cosine between document vectors \( d_i \) and \( d_j \) where \( d_i = [x_1, x_2, x_3, \ldots, x_n] \) and \( d_j = [y_1, y_2, y_3, \ldots, y_n] \) can be computed according to the cosine distance formula as mentioned in eqn (4).

\[
\cos \theta_{ij} = \frac{\sum_{k=1}^{n} x_k y_k}{\sqrt{\sum_{k=1}^{n} x_k^2 \sum_{k=1}^{n} y_k^2}} \quad \text{(4)}
\]
where \( d_i \) and \( d_j \) are the \( i^{th} \) and \( j^{th} \) document vector, \( |d_i| \) and \( |d_j| \) denotes the euclidean length (L2) of vector \( d_i \) and \( d_j \) respectively.

**K-means Algorithm**

K-means is one of the most efficient methods for clustering. It is the centroid based technique where from the given set of \( n \) data, \( k \) different clusters; each cluster characterized with a unique centroid (mean) is partitioned using the K-means algorithm. The elements belonging to one cluster are close to the centroid of that particular cluster and dissimilar to the elements belonging to the other cluster.

1. Identifying unique words form the given input documents.
2. Generation of input vectors.
   a. Term Frequency (tf)
   b. Inverse Document Frequency (idf)
   c. Term Frequency Inverse Document Frequency (tfidf)
3. Selection of similarity measure for generating similarity matrix.
   i. Cosine similarity
4. Specifying the value of \( k \), i.e. number of clusters
5. Randomly select \( k \) documents and place one of \( k \) selected documents in each cluster.
6. Place the remaining documents in the clusters based on similarity between documents and the documents present in the clusters.
7. Compute centroid of each \( k \) clusters
8. Again by using similarity measure, find the similarity between the centroids and the input documents i.e. generate similarity vector.
9. Now place the documents in the clusters based on similarity between documents and the centroids of clusters.
10. After placing all the documents in the clusters compare the previous iteration clusters with current iteration clusters.
11. If all the clusters contains same documents in previous and current iteration then terminate the algorithm here and we will be obtaining the final clusters.
12. Else repeat through step-7

**MapReduce**

MapReduce is a software framework introduced by Google to compute large scale data. It's based on functional programming paradigm with map and reduce functions. The map functions processes the input set of data and generates a set of intermediate key/value pairs. The reduce function merges the intermediate pairs with the same key. Multiple map and/or reduce tasks are run in parallel over disjoint portions of the input or intermediate data, thus parallelizing the computation. It has been hugely used inside Google for parallel-programming over clusters of computers that have unreliable communication.

**Hadoop**

Hadoop is a distributed file system written in Java with an additional implementation of Google’s MapReduce framework [3] that enables application based on map-reduce paradigm to run over the file system. It provides high throughput access to data and is suited for working with large scale data (typical block size is 64 Mb)[1].

Hadoop is an Apache software framework that analyzes petabytes of unstructured data and transforms it into a more manageable form for applications to work on. Based on Google’s MapReduce and distributed file system work, Hadoop is designed specifically to be deployed on commonly available, general-purpose network and server hardware. It is available in both open source and commercial packages [4].

### III. Experiment

For the purpose of experimentation, Clustering was done using the above data sets. For these datasets, stop-list was used to remove common words, and the words were stemmed using Porters stemming algorithm to generate unique term for clustering. Then the terms were extracted. After extracting the terms, the term frequency was generated and inverse document frequency was generated. Finally the term frequency and inverse document frequency (tfidf) was calculated which was the required term vector for calculation for cosine similarity.

After the calculation of tfidf, the k-means clustering was applied where the centroid was chosen randomly. And finally Clustering results were noted for two cases, with and without using Hadoop.

The K-means algorithm is applied on Hadoop and map reduce environment and the time utilized for the generation of various clusters size for different data sets are obtained for both environment using Hadoop and without using Hadoop.
A. Data sets

The datasets used are primarily the sample text files where the few sample text were created manually for the test. As well the datasets which are extensively used for clustering was used [5] for the work. They are heterogeneous in terms of document size, cluster size, number of classes, and document distribution. Below is the topics of the newsgroups arranged by Jason Renn [7].

Table 2. List of Topics of 20 Newsgroups

<table>
<thead>
<tr>
<th>Comp graphics</th>
<th>rec.autos</th>
<th>sci.crypt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comp.os ma- win-ux</td>
<td>rec.motorcycle</td>
<td>sci.electronics</td>
</tr>
<tr>
<td>Comp.sys ibm- pc - hardware</td>
<td>rec.sport.baseball</td>
<td>sci.med</td>
</tr>
<tr>
<td>Comp.sys misc. hardware</td>
<td>rec.sport.hockey</td>
<td>sci.space</td>
</tr>
<tr>
<td>Comp.windows.x</td>
<td>talk.politics.misc</td>
<td>talk.religion.misc</td>
</tr>
<tr>
<td>Misc. forsale</td>
<td>talk.politics.guns</td>
<td>alt.atheism</td>
</tr>
<tr>
<td>Talk.politics.misc</td>
<td>soc.religion.christian</td>
<td></td>
</tr>
</tbody>
</table>

B. Tools

The experiment was performed under the system consisting of following hardware configuration and software system.

i. Hardware Configuration

The hardware configuration used for the experiment are RAM:4GB, Processor: Intel® Core(TM) i7-2670QM CPU@2.20 GHz, System type:64 bit operating system/Linux(UBUNTU 12.04).

ii. Software

The software used for the experiment are as follows:

<table>
<thead>
<tr>
<th>S. No</th>
<th>Software</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Operating System (OS)</td>
<td>3.2.0-23-generic</td>
</tr>
<tr>
<td>2</td>
<td>Java platform</td>
<td>Intellij IDEA 12.0</td>
</tr>
<tr>
<td>3</td>
<td>Apache Hadoop</td>
<td>2.2.0</td>
</tr>
<tr>
<td>4</td>
<td>Apache Mahout</td>
<td>0.9</td>
</tr>
<tr>
<td>5</td>
<td>Apache Maven</td>
<td>3.0.5</td>
</tr>
<tr>
<td>6</td>
<td>Apache Lucene</td>
<td>2.4.0</td>
</tr>
</tbody>
</table>

IV. RESULT AND ANALYSIS

In order to know the performance for k-means clusters generated with and without using Hadoop, the time taken was observed and compared. Similarly the distributed system over centralized system was also analysed on the basis of time utilized during the execution of program for obtaining output with Hadoop and without Hadoop.

For the performance evaluation, the time used by the datasets while generating clusters 2,3,4,5,6,7,8,9,10,20,40 and 60 were observed. The time was observed each time for the mentioned datasets 100, 500, 1000, 1500, 2000, 2500, 3500 and 20000 for the case with Hadoop and without Hadoop while generating different clusters ranging from 2,3,4,5,6,7,8,9,10,20,40 and 60. The datasets are the size of 244.8 KB, 1 MB, 2.9 MB, 3.9 MB, 4 MB, 5.1 MB, 5.4 MB, 6 MB, 6.4 MB, 46.4 MB respectively for the datasets 100, 500, 1000, 1500, 2000, 2500, 3000, 3500 and 20000. The time was observed in seconds. The obtained values for time for each datasets with Hadoop and without Hadoop are used for evaluation of the performance. So obtained time is compared for the clusters 2,3,4,5,6,7,8,9,10,20,40 and 60 for each datasets.

The value observed for time for the cases with and without Hadoop are listed in the table 4 for all the datasets 100, 500, 1000, 1500, 2000, 2500, 3000, 3500 and 20000 generating clusters 2,3,4,5,6,7,8,9,10,20,40 and 60 using Hadoop and without using Hadoop. The radar chart are obtained for all mentioned clusters and datasets. But here radar chart for clusters 2, 8, 20 and 60 are shown in the figures 2, 3, 4 and 5.

The time chart obtained for clusters 2, 3, 4, 5, 6, 7, 8, 9, 10 and 20, 40 and 60 showing time for the different datasets having different data size with and without Hadoop are shown in table no 4. For all clusters obtained it can be seen that time taken with Hadoop is comparatively less than without using Hadoop which can be clearly seen in figure no 2,3,4 and 5.

Viewing with respective to the datasets of different size without Hadoop it is clearly seen that as the size of data increases the time utilization also increases. The greater the size of data the more time it takes to execute that is to generate the cluster which can be seen in figure no 2,3, 4 and 5.

For the 100 datasets of size 244.8 KB the time taken for generating clusters 2 to 10 without Hadoop is comparatively less than with using Hadoop. As the data size increases, for generating clusters without Hadoop the time taken increases gradually whereas with Hadoop time taken almost remain constant. While generating clusters 2 to 10 for 1500 datasets of size 3.9 MB and 3500 datasets of size 6.4 MB without Hadoop time taken increases in comparison to 100 datasets whereas with Hadoop time remains almost constant for the datasets 100, 1500 and 3500. But as the size of data increases time utilization is higher without Hadoop for the generation of clusters than with Hadoop which is clearly shown in radar chart 2,3,4,5. For the dataset 20000 of size 46.6 MB without using Hadoop, during generation of clusters the time cannot be observed due to the memory error. But with Hadoop time observed for the 20000 datasets is shown in figure 2,3,4,5. This clearly shows that the time taken to obtain clusters for large datasets with Hadoop is far more less than without Hadoop.

Without using Hadoop for dataset 100 of size 233.4 KB it takes lesser time to execute that is about 2 sec. As the size of data increases the time consumption also
increases gradually. For the datasets 3500 of size 6.4 MB it takes more time in comparison to other datasets. It is similar to all other clusters obtained from 2 to 10 and is for the clusters obtained for 20, 40 and 60. So as the number of datasets increases that is size of data increases the time taken also increases that is data size is directly proportional to the time. That is why for the dataset 20000 of size 46.4 MB the time couldn’t be measure due memory error which is shown in figure no 2, 3, 4 and 5. Using Hadoop the time for clusters 2 to 10 are almost same. In this case for small datasets the time is almost same but for the dataset 20000 of size 46.4 MB it is comparatively Hence through radar chart it is seen that for k-means algorithm the time taken to generate clusters without Hadoop is more with Hadoop.

It can be summarized that with the increment in the size of data, Without using Hadoop, the time utilization rises as well whereas with using Hadoop, time taken for all datasets used is almost same.

For number of documents greater than 3500, without using Hadoop, while generating clusters 2-10, out of memory error is displayed. Hence time and memory couldn’t be observed for those cases. But when using Hadoop, clustering can be easily generated within the available system resources. The time taken for the execution of clusters without using Hadoop for small scale documents is low in comparison to the time taken in Hadoop.

<table>
<thead>
<tr>
<th>Table 4 Tabular chart for time analysis observed for different datasets with and without Hadoop.</th>
</tr>
</thead>
<tbody>
<tr>
<td>No of Clusters</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>40</td>
</tr>
</tbody>
</table>
Datasets

<table>
<thead>
<tr>
<th>No of Clusters</th>
<th>Hadoop</th>
<th>100 (244.8 KB)</th>
<th>500 (1 MB)</th>
<th>1000 (2.9 MB)</th>
<th>1500 (3.9 MB)</th>
<th>2000 (4 MB)</th>
<th>2500 (5.1 MB)</th>
<th>3000 (5.4 MB)</th>
<th>3500 (6.4 MB)</th>
<th>20000 (6.4 MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>With Hadoop</td>
<td>66</td>
<td>128</td>
<td>81</td>
<td>301</td>
<td>312</td>
<td>87</td>
<td>141</td>
<td>123</td>
<td>204</td>
<td></td>
</tr>
<tr>
<td>Without Hadoop</td>
<td>2</td>
<td>5</td>
<td>157</td>
<td>277</td>
<td>208</td>
<td>436</td>
<td>315</td>
<td>Out of memory</td>
<td>Out of memory</td>
<td></td>
</tr>
<tr>
<td>With Hadoop</td>
<td>69</td>
<td>158</td>
<td>89</td>
<td>339</td>
<td>311</td>
<td>98</td>
<td>134</td>
<td>131</td>
<td>143.4</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2. Radar chart showing time obtained for the different datasets for cluster 2 with and without using Hadoop

Figure 3. Radar chart showing time obtained for the different datasets for cluster 8 with and without using Hadoop

Figure 4. Radar chart showing time obtained for different datasets including 20000 datasets for cluster 20 with and without using Hadoop.

Figure 5. Radar chart showing time obtained for the datasets including 20000 datasets for cluster 60 with and without using Hadoop.

As the data size increases the time utilization also increases gradually and it becomes infinite for the data sets 3500 and above for the case without using Hadoop. But with using Hadoop the time taken is comparatively less than that of without using Hadoop for the data sets 3500 and above. Here using Hadoop for data sets 1500 and 2000 time taken is slightly more than the case without using Hadoop.

But It was also noticed that in the used experiment environment with the increase in number of datasets and the cluster results in increase of the execution time both with and without using Hadoop.

V. CONCLUSIONS AND FUTURE WORK

In this work, new model for document clustering was given which can be used to organize In this work, new model for document clustering was given which can be used to organize documents into sub-folders without having to know about the contents of the documents.
This improves the performance of information retrieval in any scenario. To scale the document clustering the proposed model uses the MapReduce implementation of k-means from Apache Hadoop Project.

Hadoop shows the high performance for document clustering using k-means algorithm. Hadoop works with large datasets making efficient utilization of time, memory and processor. Without Hadoop the size of data is directly proportional to the time, memory and processor utilization. As the size of data increases the utilization in time, memory and processor increases leading to the excessive use of system resources resulting low performance and inefficiency to the system.

Hadoop works with the MapReduce and HDFS architecture which implements the k-means algorithm for large datasets of size 46.4 MB within the defined system resources in comparatively less consumption of time, memory and processor representing the distributed system. In other hand the case without Hadoop resembles the centralized system with higher consumption of system resources.

Since the system configuration used was insufficient for case without Hadoop, the results can be tested on the higher configurable system. The clustering has been done for the files and folders consisting of text files. Similar work can be carried on for the different file format multimedia files, videos, images, audios and so on. The indexing has can be done using apache tikka and other application as solitary. Since the term vector can generate sparse matrix that decreases the efficiency so further enhancement can be done.

REFERENCES


