TEXT MINING WITH LUCENE AND HADOOP:

DOCUMENT CLUSTERING WITH FEATURE EXTRACTION

BY

DIPESH SHRESTHA

A THESIS SUBMITTED FOR THE FULFILLMENT

OF

RESEARCH DEGREE

TO

WAKHOK UNIVERSITY

2009

ABSTRACT

In this work a new method for document clustering has been adapted using non-negative matrix factorization. The key idea is to cluster the documents after measuring the proximity of the documents with the extracted features. The extracted features are considered as the final cluster labels and clustering is done using cosine similarity which is equivalent to k-means with a single turn.

An application was developed using apache lucene for indexing documents and map-reduce framework of apache hadoop project was used for parallel implementation of k-means algorithm from apache mahout project. This application was named 'Swami'. The performance of models with proposed technique was conducted on news from 20 newsgroups datasets and the accuracy was found to be above 80% for 2 clusters and above 75% for 3 clusters. Since the experiments were carried only in one cluster of hadoop, the significant reduction in time was obtained by map-reduce implementation when clusters size exceeded 9 i.e. 40 documents averaging 1.5 kilobytes.

Thus it's concluded that the feature extracted using NMF can be used to cluster documents considering them to be final cluster labels as in k-means, and for large scale documents the parallel implementation using map-reduce can lead to reduction of computational time.

ACKNOWLDEGEMENTS

I would like to thank my advisor professor Andoh Tomoharu for giving me the opportunity to work on this project and his valuable guidance. I would also like to thank ex-president of this university, Maruyama Fujio for building my interest in map-reduce, professor Numata Yasuhide for his valuable help in understanding mathematical formulas. I am grateful to Bishnu Prasad Gautam for being my supervisor, friends in apache mahout and apache lucene project for their help in using the libraries. My thank goes to Bishal Acharya for all his support. Finally, I am very indebted to Swami for always being with me, my sister Kripa and my parents for their constant support and encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The need for the organisation of data is a must for a quick and efficient retrieval of information. A robust means for organisation of data in any organisation has been the use of databases. Databases like the relational, object-oriented or object-relational databases, all have well structured format to keep data. Not all information that an organisation generates is kept or can be kept in databases. Information are stored in the huge amount in form of unstructured or semi-structured documents. Organising these documents into meaningful groups is a typical sub-problem of Information Retrieval, in which there is need to learn about the general content of data, Cutting D et al. [1].

1.1 Motivation

The primary structure for organisation of computer files are placing them into folders and placing the folders again into some higher level folders. To place these files into folders manually, information about the content of the files are needed. Typically the name of file is enough to give impression of the contents of the files accordingly to which the files can be grouped together. There are certain instances in which it becomes difficult to manually group the files, for instance when they are in huge number, when their contents can't be distinguished from their names. This is where there is an ardent need of computer aided clustering of the documents.

Recently there has been surge of interest in document clustering after Lee and Sung's [2,3] update rules for NMF proved to perform better than Latent Semantic

Indexing (LSI) with Singular Value Decomposition (SVD). Many researchers are approaching with efficient algorithms and comprehensive comparison of the existing algorithms [20,35] but, still, these have been limited to academics. In this work a new working model based on Lee and Sung's [3] update rules for NMF is presented for automatic document clustering with an application developed for its implementation.

For the experimentation purpose of this model, the data set from Newgroup 20 was used. To aid to NMF, removal of common clutter/stop-words using key-words from Key Phrase Extraction Algorithm [6] and stemming from Porter algorithm [5] has been utilised in pre-processing step. Finally, to study the performance of the map-reduce framework in Hadoop the parallel implementation of k-means clustering algorithm has been used.

The performance of models with proposed technique was conducted and found to be above 80% for 2 clusters and above 75% for 3 clusters. The time taken was reduced by map-reduce implementation when clusters size exceeded 9 i.e. 40 documents averaging 1.5 kilobytes.

1.2 The organisation of the thesis

The remaining part of the thesis is organised as follows:

Chapter 2: This chapter gives the background foundation of this work and the related works in the field of document clustering.

Chapter 3: This chapter discusses the proposed model in details and also introduces concepts of map-reduce and Apache Hadoop.

Chapter 4: In this chapter design and implementation of the proposed application is discussed. It includes UML diagrams and interfaces diagrams.

Chapter 5: The experiment on the proposed model using the application is explained here. Also the results are shown in this chapter.

Chapter 6: This chapter summarises the work with conclusion from the experiments performed in Chapter 5.

Chapter 7: Finally the future works recommendation for this work in presented in this chapter.

CHAPTER 2

BACKGROUND AND RELATED WORK

The following section gives background information on text mining, supervised and unsupervised learning, document clustering, Vector Space Model, feature extraction and NMF. It finally shows how NMF has been used for document clustering.

2.1 Text mining

"Sifting through vast collections of unstructured or semi-structured data beyond the reach of data mining tools, text mining tracks information sources, links isolated concepts in distant documents, maps relationships between activities, and helps answer questions."

-Tapping the Power of Text Mining , Communications of the ACM, Sept. 2006

Text mining is a mechanism to understand and extract meaningful implicit information from large amount of the semi-structured or unstructured text data. This encompasses the combination of human linguistic capacity and computational power of computers. The linguistic capacity includes the ability to differentiate spelling variations, filter out noisy data, understanding the synonyms, slangs and abbreviations, and finding the contextual meaning. The computational power of computers include the ability to statistical/probabilistic processing of large volume at high speed. Some application of text mining are: information extraction, topic detection and tracking, summarization, categorization, clustering, concept linkage, information visualization etc. Broadly the text mining algorithms can be categorized into supervised learning and unsupervised learning.

2.1.1 Supervised learning

Supervised learning is a technique in which the learning process is supervised by correct data before the prediction is made on the targets. It consists of finding the relationship between the predictor and target attribute. If the algorithm can predict a categorical value for a target attribute, it is called a classification function. Class is an example of a categorical variable which does not have partial ordering. For example, labelling a customer as a possible buyer or not a possible buyer can be a classification problem. If the algorithm can predict a numerical value then it is called regression. Unlike the categorical values numerical values have partial ordering. For example, forecasting the temperature the next day, estimating the profit/loss for the next term etc.

2.1.2 Unsupervised learning

Unlike the supervised learning this technique doesn't require training to yield output. It only uses the predictor attribute values to gain understanding of the structure and relationship of the data. Finding out the number of market segments, determining the themes of news etc. can be examples of unsupervised learning. Algorithms under unsupervised learning are feature extraction, clustering, association rule mining, density estimation etc.

2.2 Document clustering

Document clustering can loosely be defined as “clustering of documents”. Clustering is a process of understanding the similarity and/or dissimilarity between the given objects and thus, dividing them into meaningful subgroups sharing common characteristic. Good clusters are those in which the members inside the cluster have quite

a deal of similar characteristics. Since clustering falls under unsupervised learning, predicting the documents to fall into certain class or group isn't done. The methods of document clustering can be categorized into two groups;

a. Document partitioning (Flat Clustering)

This approach divides the documents into disjoint clusters. The various methods in this category are : k-means clustering, probabilistic clustering using the Naive Bayes or Gaussian model, latent semantic indexing (LSI), spectral clustering, non-negative matrix factorization (NMF).

b. Hierarchical clustering

This approach finds successive clusters of document from obtained clusters either using bottom-up (agglomerate) or top-bottom (divisive) approach.

2.3 Vector Space Model (VSM)

Various mathematical models have been proposed to represent Information Retrieval systems and procedures; the boolean model, the probabilistic model, the vector space model etc. Of the above models, the vector space model has been quite widely used model.

2.3.1 Term-document matrix and term weighting

In VSM, a collection of d documents described by t terms can be represented as a t×d 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. Each component of the document vector reflects a particular term which may or may not be in the document. The value of each component depends on the degree of relationship between its associated term and the respective document, commonly called term weighting. As the Vector Space Model requires that the relationship be described by a single numerical value, let a_{ij} represent the degree of relationship between term i and document j.

a. Binary weighting

This is the simplest term weighting method where if $a_{ij}=1$ means term i occurs in document j, $a_{ij}=0$ meaning otherwise. 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.

b. Term frequency weighting

In this scheme $a_{ij} = tf_{ij}$ where tf_{ij} denotes how many times term i occurs in document j. This scheme is more informative than the binary weighting but, nevertheless it suffers from shortcoming as it focuses only on the local weight, neglecting the global weight.

c. Term frequency-inverse document frequency weighting

This schemes overcomes the drawback of term frequency model by including the global weight. To understand the global weight and the local weight following table can be used.

	Doc1	Doc2	Doc3	Doc4
Term1	2	1		3
Term2		2		1

Table 2.1: Term-document Matrix

The term1 occurs in total of 3 document while term2 occurs in total of 2 document. Now, the global weight of term1 = number of document(s) with the term1/total documents = log(4/3) = 0.124. Similarly, the global weight of term2 = log(4/2) = 0.30. Thus, the global weight is the overall importance of the term which decreases as the number of document containing the term increases. The tf-idf scheme aims at balancing the local and the global term occurrences in the documents and can be defined as, $a_{ij}=tf_{ij} \cdot \log(N/df_i)$ where tf_{ij} is the term frequency in document d_j, df_i denotes the number of documents in which term i appears, and N represents the total number of documents in the collection. In this work tf-idf was adapted as standard term weighting scheme.

2.3.2 Common Measure of Similarity in VSI

2.3.1 Euclidean

This is the most commonly used measure of similarity between two vectors. Let there be two vectors d_i and d_j where d_i ={x1,x2,x2...xn} and d_j = {y1,y2,y3...yn}, then it measures the distance between the vectors as

$$D=\sqrt{\sum_{k=1}^{n}\left(x_k-y_k\right)^2} \qquad (1)$$

As the distance increases between the vectors increases, the vectors are said to be dissimilar.

2.3.2 Cosine

It's a common measure of similarity between two vectors which measures the cosine of the angle between them. In a t×d term-document matrix A, the cosine between document

vectors d_i and d_j where d_i ={x1,x2,x2...xn} and d_j = {y1,y2,y3...yn} can be computed according to the cosine distance formula:

$$\cos\theta_{i,j} = \frac{d_i \cdot d_j}{|d_i||d_j|} = \frac{\sum_{k=1}^{n} x_k \cdot y_k}{|\sqrt{\sum_{k=1}^{n} x_k^2 \cdot \sum_{k=1}^{n} y_k^2}|} \quad (2)$$

where d_i and d_j are the i^{th} and j^{th} document vector, $|d_i|$ and $|d_j|$ and denotes the euclidean length (L_2) of vector d_i and d_j respectively. The greater the value of equation (2) , the more similar they are said to be. As multiplying the vectors does not change the cosine, the vectors can be scaled by any convenient value. Very often the document vectors are normalised to a unit length. In high dimensionality such as VSI, the cosine measure has shown better performance than Euclidean measure of similarity [14].

2.4 Feature extraction

Traditional methods in document clustering use words as measure to find similarity between documents. These words are assumed to be mutually independent which in real application may not be the case. Traditional VSI uses words to describe the documents but in reality the concepts/semantics/features/topics are what describes the documents. The extraction of these features from the documents in Feature Extraction. The extracted features hold the most important idea/concept pertaining to the documents. Feature extraction has been successfully used in text mining with unsupervised algorithms like Principal Components Analysis (PCA), Singular Value Decomposition (SVD), and Non-Negative Matrix Factorization (NMF) involving factoring the document-word matrix[8].

2.5 Latent Semantic Indexing (LSI)

Latent Semantic Indexing (LSI) is a novel Information Retrieval technique that was designed to address the deficiencies of the classic VSM model:

"LSI tries to overcome the problems of lexical matching by using statistically derived conceptual indices instead of individual words for retrieval. LSI assumes that there is some underlying or latent structure in word usage that is partially obscured by variability in word choice. A truncated Singular Value Decomposition (SVD) is used to estimate the structure in word usage across documents. Retrieval is then performed using a database of singular values and vectors obtained from the truncated SVD. Performance data shows that these statistically derived vectors are more robust indicators of meaning than individual terms." Berry et. al [70]

Application of Latent Semantic Indexing with results can be found in Berry et al. [20] and Landauer et al in [21].

2.5.1 Limitations of LSI and superiority of NMF

Singular Value Decomposition is widely used in standard factorization of data matrix. Using SVD the vectors from the feature space contain negative values. Since every data vector in VSM has positive value but the newly found vectors from the feature space contain negative values, it makes difficult for interpretation. This is not the case with NMF. NMF has been noted to have advantages over standard PCA/SVD based factorization in paper by Lee and Seung [2,3]. In contrast to cancellations due to negative entries in matrix factors in SVD based factorizations, the non-negativity in NMF ensures factors contain coherent parts of the original data (text,images etc.)[11]

2.6 Non-negative matrix factorization (NMF)

Non-negative matrix factorization is a special type of matrix factorization where the constraint of non-negativity is on the lower ranked matrices. It decomposes a matrix V_{mn} into the product of two lower rank matrices W_{mk} and H_{kn}, such that V_{mn} is approximately equal to W_{mk} times H_{kn}.

$$V_{mn} \approx W_{mk} \cdot H_{kn} \tag{3}$$

where, k << min(m,n) and optimum value of k depends on the application and is also influenced by the nature of the collection itself [18]. In the application of document clustering, k is the number of features to be extracted or it may be called the number of clusters required. V contains column as document vectors and rows as term vectors, the components of document vectors represent the relationship between the documents and the terms. W contains columns as feature vectors or the basis vectors which may not always be orthogonal (for example, when the features are not independent and have some have overlaps). H contains columns with weights associated with each basis vectors in W.

Thus, each document vector from the document-term matrix can be approximately composed by the linear combination of the basis vectors from W weighted by the corresponding columns from H. Let v_i be any document vector from matrix V, column vectors of W be $\{W_1, W_2,...,W_k\}$ and the corresponding components from column of matrix H be $\{h_{i1},h_{i2},...,h_{ik}\}$ then,

$$v_i \approx W_1 \cdot h_{i1} + W_2 \cdot h_{i2} + \ldots + W_k \cdot h_{ik} \tag{4}$$

NMF uses an iterative procedure to modify the initial values of W_{mk} and H_{kn} so

that the product approaches V_{mn}. The procedure terminates when the approximation error converges or the specified number of iterations is reached. The NMF decomposition is non-unique; the matrices W and H depend on the NMF algorithm employed and the error measure used to check convergence. Some of the NMF algorithm types are, multiplicative update algorithm by Lee and Seung [3], sparse encoding by Hoyer [15], gradient descent with constrained least squares by Pauca [16] and alternating least squares algorithm by Pattero [17]. They differ in the measure cost function for measuring the divergence between V and WH or by regularization of the W and/or H matrices.

Two simple cost functions studied by Lee and Seung are the squared error (or Frobenius norm) and an extension of the Kullback-Leibler divergence to positive matrices. Each cost function leads to a different NMF algorithm, usually minimizing the divergence using iterative update rules. Using the Frobenius norm for matrices, the objective function or minimization problem can be stated as

$$\min_{W,H} \|V - WH\|_F^2 \qquad (5)$$

where W and H are non-negative. The method proposed by Lee and Sung [3] based on multiplicative update rules using Forbenus norm, popularly called multiplicative method (MM) can be described as follows.

2.6.1 MM Algorithm

(1) Initialize W and H with non-negative values.

(2) Iterate for each c, j, and i until within approximation error converges or after l iterations:

(a) $$H_{cj} \leftarrow H_{cj} \frac{\left(W^T V\right)_{cj}}{\left(W^T WH\right)_{cj} + e} \quad (6)$$

(b) $$W_{ic} \leftarrow W_{ic} \frac{\left(VH^T\right)_{ic}}{\left(WHH^T\right)_{ic} + e} \quad (7)$$

In steps 2(a) and (b), e, a small positive parameter equal to 10^{-9} , is added to avoid division by zero. As observed from the MM Algorithm, W and H remain non-negative during the updates.

Lee and Seung [3] proved that with the above update rules objective function (3) achieve monotonic convergence and is non-increasing, they becomes constant if and only if W and H are at a stationary point. The solution to the objective function is not unique.

2.7 Document clustering with NMF

Ding C et. al in [11] shown that when Frobenius norm is used as a divergence and adding an orthogonality constraint $H^T H = I$, NMF is equivalent to a relaxed form of k-means clustering. Wei Xu et al. were the first ones to use NMF for document clustering in [9] where unit euclidean distance constraint was added to column vectors in W. Yang et al. [10] extended this work by adding the sparsity constraints because sparseness is one of the important characters of huge data in semantic space. In both of the work the clustering has been based on the interpretation of the elements of the matrices.

"There is an analogy with the SVD in interpreting the meaning of the two non-negative matrices U and V. Each element u_{ij} of matrix U represents the degree to which term $f_i \in W$ belongs to cluster j, while each element v_{ij} of matrix V indicates to which degree document i is associated with cluster j. If document i solely belongs to cluster x, then v_{ix}

will take on a large value while rest of the elements in i^{th} row vector of V will take on a small value close to zero. "[9]

In the above statement, $W \approx UV$

From the work of Kanjani K [12] it is seen that the accuracy of algorithm from Lee and Seung [3] is higher than their derivatives [9,10]. In this work, the original multiplicative update proposed by Lee and Seung in [3] is undertaken.

CHAPTER 3

METHODOLOGY

The following section describes the proposed model in details. It includes the clustering method with the proposed model and the parallel implementation strategy of k-means. The latter parts contains explanation on the k-means algorithm, brief introduction to map-reduce and underlying architecture of Hadoop Distributed File System (HDFS) used to run k-means algorithm.

3.1 The Proposed Model

From hereafter called KNMF. In KNMF the document clustering is done on basis of the similarity between the extracted features and the individual documents. Let extracted feature vectors be $F=\{f_1,f_2,f_3....f_k\}$ computed by NMF. Let the documents in the term-document matrix be $V = \{d_1,d_2,d_3....d_n\}$, then document d_i is said to belong to cluster f_x if, the angle between d_i and f_x is minimum.

3.1.1 The methodology adapted

1. Construct the term-document matrix V from the files of a given folder using term frequency-inverse document frequency
2. Normalize length of columns of V to unit Euclidean length.
3. Perform NMF based on Lee and Seung [3] on V and get W and H using (3)
4. Apply cosine similarity to measure distance between the documents d_i and extracted features/vectors of W. Assign d_i to w_x if the the angle between d_i and w_x is smallest. This is equivalent to k-means algorithm with a single turn.

To run the parallel version of k-means algorithm, Hadoop is started in local reference mode and pseudo-distributed mode and the k-means job is submitted to the JobClient. The time taken for steps from 1 though 3 and the total time taken were noted separately.

3.1.2 Steps in Indexing the documents in a folder

1. Determine if the document is new, update in index of not updated in index.
2. If it's up to date then do nothing what follows. If the document is new, create a Lucene Document, if it's not updated then delete the old document and create new Lucene Document.
3. Extract the words from the document.
4. Remove the stop-words.
5. Apply Stemming.
6. Store the created Lucene Document in index.
7. Remove stray files.

The Lucene Document contains three fields: path, contents and modified which respectively stores the full-path of the document, the terms and modified date(to seconds). The field *path* is used to uniquely identify documents in the index, the field *modified* is use to avoid re-indexing the documents again if it's not modified. In the step 7) the documents which have been removed from the folder but with entries in the index are removed from index. This step has been followed to keep the optimal word dictionary size.

The default stop-words were added from the project of Key Phrase Extraction Algorithm [6] which defines some 499 stop-words. The stop-words are read from text file and users can add words to the text file. After the removal of stop words, the document was stemmed by the Porter algorithm [4].

3.2 Parallel implementation of k-means

The parallel implementation strategy of k-means algorithms in multi-core is described in [26] as:

"In k-means [12], it is clear that the operation of computing the Euclidean distance between the sample vectors and the centroids can be parallelized by splitting the data into individual subgroups and clustering samples in each subgroup separately (by the mapper). In recalculating new centroid vectors, we divide the sample vectors into subgroups, compute the sum of vectors in each subgroup in parallel, and finally the reducer will add up the partial sums and compute the new centroids. "

In the same paper it was noted that the performance of k-means algorithm with map-reduce increased in an average 1.937 times than than its serial implementation without map-reduce. From as low as 1.888 times in Synthetic Time Series (sample = 100001 and features = 10) to as high as 1.973 times in KDD Cup 999 (sample = 494021 and features = 41). It also adds that it was possible to achieve 54 times speedup on 64 cores.

Indeed, the performance upgrading with the increase of number of cores was

almost linear. This paper was the source for the foundation of Mahout project[1] which include the implementation strategy of k-means algorithm in map-reduce over the Hadoop. Implementation of k-means in MapReduce is also presented in lectures from [23]. Drost, I [22] describes k-means of Mahout project. In [24] Gillick et. Al studied the performance of Hadoop's implementation of MapReduce and have suggested performance enhancing guidance as well.

3.2.1 MapReduce in KNMF

In the method proposed in this work, since the final clusters are the features extracted from the NMF algorithms, the parallelization strategy of map-reduce can be applied to compute the distance between the data vectors and the feature vectors. Since it requires only one iteration, it can be considered as having only one map-reduce operation. Furthermore, since the cluster centroid computation isn't needed only one map operation is enough. The map operation intakes the list of feature vectors and individual data vectors and outputs the closest feature vector for the data vector.

For instance, we have list of data vectors V = {v1,v2,...vn} and list of feature vectors W= {w1,w2,w3} computed by NMF. Then,

$$<v_i, W> \rightarrow \text{map} \rightarrow <v_i, w_x>$$

where w_x is the closest (cosine similarity,see section 2.3.2) feature vector to data vector v_i.

1 http://lucene.apache.org/mahout/

3.3 K-Means Algorithm

3.3.1 Standard k-means

It's one of the oldest and most widely used algorithm for clustering data sets. K-means algorithm is an expectation maximization algorithm that tries to find the cluster centers. It tires to minimize the overall inter-cluster variance, or, the squared error function.

$$V=\sum_{i=1}^{k}\sum_{x_j\in S_i}\left(x_j-\mu_i\right)^2 \qquad (8)$$

where there are k clusters S_i, i = 1, 2, ..., k, and μ_i is the centroid or mean point of all the points $x_j \in S_i$. With k-means, the number of clusters should be defined prior to running the algorithm. The complexity of this algorithm is O(t.d.k.m) where d is number of documents, t is the number of terms, k is the clusters required and m is the maximum number of iteration [14].

The basic algorithm is to group points into the nearest of the pre-defined number of clusters until the updated cluster center is within some tolerance of old respective cluster center or until a certain number of iterations has been achieved. More specifically the following pseudo code can represent the k-means algorithm.

/** Pseudo code for Kmeans **/

Select any k number of cluster centers c_i (i=1,2..k) and define convergence tolerance t.

Until x number of iterations is reached

For each point, p_j (j=1,..n) find its nearest cluster.

If all the new cluster centers c_i, are within tolerance t of its old cluster centers then stop else update to new cluster centers.

/** Pseudo code for Kmeans **/

An excellent tutorial on k-means can be reached in [7].

3.3.2 Spherical k-means(k-means on a unit hypersphere)

For high-dimensional data such as text data represented in document-term matrix, cosine similarity has been shown to be more superior to Euclidean distance[14]. Since the vectors with same component but different totals are treated identically, the direction is important than the magnitude and the document vectors can be normalized to unit sphere for more efficient processing [19]. The spherical k-means algorithm tries to maximize the average cosine similarity objective:

$$L=\sum_{i=1}^{k}\sum_{x_j\in S_i} x_j^T \mu_i \qquad (9)$$

where there are k clusters S_i, $i = 1, 2, ..., k$, and μ_i is the centroid or mean point of all the points $x_j \in S_i$. In this work this methodology was adapted.

3.4 MapReduce

MapReduce is a software framework introduced by Google [25] to computer 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. Since the framework handles the complexity underlying the parallel

communication, it allows programmers to write functional-style code that can be automatically parallelized and scheduled in a distributed system.

MapReduce has been studied in multi-core and multi-processor system by Ranger et al. in [27] and found to achieve good scalability using Phoenix, a programming API and runtime based on Google's MapReduce. Chu et al. [26] studied the parallel implementation of machine learning algorithms in a light weight architecture for multicores based on Google's MapReduce.

3.5 Hadoop

Hadoop is a distributed file system written in Java with an additional implementation of Google's MapReduce framework [25] 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)

3.5.1 Hadoop Distributed File System (HDFS)[2]

It is its native file system that's build to with stand fault and is designed to be deployed on low-cost hardware. It's based on master/slave architecture. The master nodes are called namenodes. Every cluster has only one namenode. It manages the filesystem namespace and access to files by client (opening, closing, renaming files) . It determine the files maping blocks to slaves or datanode . Usually there is one datanode per node. The task of datanode is to manage the data stored in the node (each file is stored in one of more blocks) . It's responsible for read/write requests from clients (creation, deletion,

2 http://hadoop.apache.org/core/

replication of blocks).

All HDFS communication protocols are layered on top of the TCP/IP protocol. Files in HDFS are write-once(read many) and have strictly one writer at any time. The blocks of a file are replicated for fault tolerance. The block size and replication factor are configurable per file. Hadoop can be run in any of the below modes.

Local (Stand-alone mode)

By default, Hadoop is configured to run in a non-distributed mode, as a single Java process. This is useful for debugging.

Pseudo-distributed Mode

Hadoop can also be run on a single-node in a pseudo-distributed mode where each Hadoop daemon runs in a separate Java process.

Fully-distributed Mode

In this mode, multiple instances of Hadoop are run across multiple-nodes with distinction between master and slave nodes.

3.5.2 MapReduce framework in Hadoop[3]

The input and output to the map-reduce application can be shown as follows:

(input) <k1,v1> → map → <k2,v2> → reduce <k3,v3> (output)

The input data is divided and processed in parallel across different

3 http://hadoop.apache.org/core/docs/current/mapred_tutorial.html

machines/processes in map phase and the reduce combines the data according the key to give final output. For this sort of task the framework should be based on master/slave architecture. Since HDFS is itself based on master/slave architecture, MapReduce framework fits well in Hadoop. Moreover usually the compute nodes and the storage nodes are the same, that is, the Map/Reduce framework and the distributed filesystem are running on the same set of nodes. This configuration allows the framework to effectively schedule tasks on the nodes where data is already present, resulting in very high aggregate bandwidth across the cluster.

To implement MapReduce framework in Hadoop, there is a single master called JobTracker per job . Job is the list of task submitted to the MapReduce framework in Hadoop. The master is responsible for scheduling the jobs' component tasks on the slaves, monitoring them and re-executing the failed tasks. There can be one slave or tasktracker per cluster-node. The slaves execute the tasks as directed by the master.

CHAPTER 4

ARCHITECTURAL DESIGN/IMPLEMENTATION

This section describes the design and implementation of the application *Swami* that was created in during this work.

4.1 Information

Version	1.0
Programming Language	Java
Platform	Only tested in GNU/Linux
IDE	NetBeans IDE 6.0.1
UML Documentation	Umbrello UML Modeller 2.0.3
License	GNU GPL Version 3.0
Website	www.wakhok.ac.jp/~dipesh/swami

Table 4.1 : Information about Software

Since Hadoop, Lucene and Mahout are built with Java natively, it would be easy for interoperability between the components developed with Java. Considering this fact, Java was chosen as the programming language. Umbrello UML Modeller has a simple yet powerful set of modelling tools, due to which it was used for UML Documentation. NetBeans IDE was chosen as developmental IDE accounting to its rich set of features and easy GUI Builder tool. Respecting the principles of freedom asserted by the open source movement, this application *Swami* is licensed under GNU GPL Version 3.0 which can be obtained from the above mentioned website.

4.2 Use case Diagrams

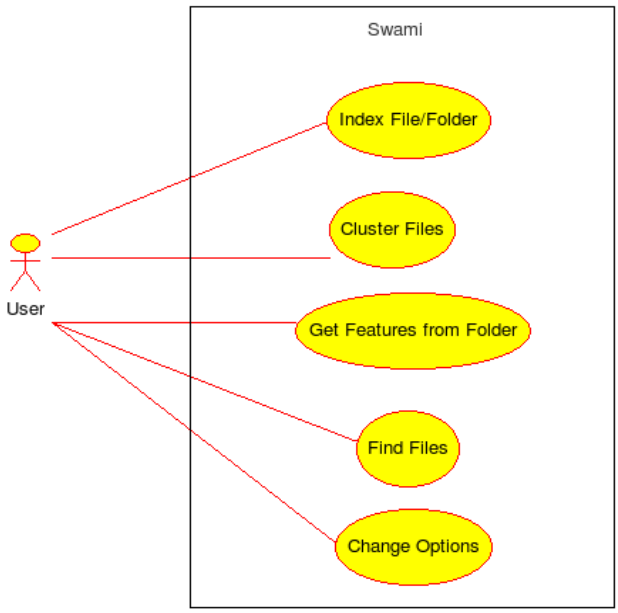


Figure 4.1: The Use Case Diagram

Users of Swami interact with the system according to the above shown cases. Basically users index files/folders and cluster files. Users can choose how to perform clustering. It can be done with/without using NMF and/or with/without using Hadoop. Besides, these two major use cases, users can get the main features from the folder, for this again NMF has been used and features are shown by the words with maximum weights in matrix W. (see section 2.6 for matrix information). Users also can find the files they are looking for if they know the date of modification. Files are show according to the extracted feature ordered by the weights in matrix H. Users have the option to change parameters like number of words to represent each feature, number of files to be shown under the features, NMF parameters like convergence/iterations, file for stop-words, location of index, location of folders to be indexed etc.

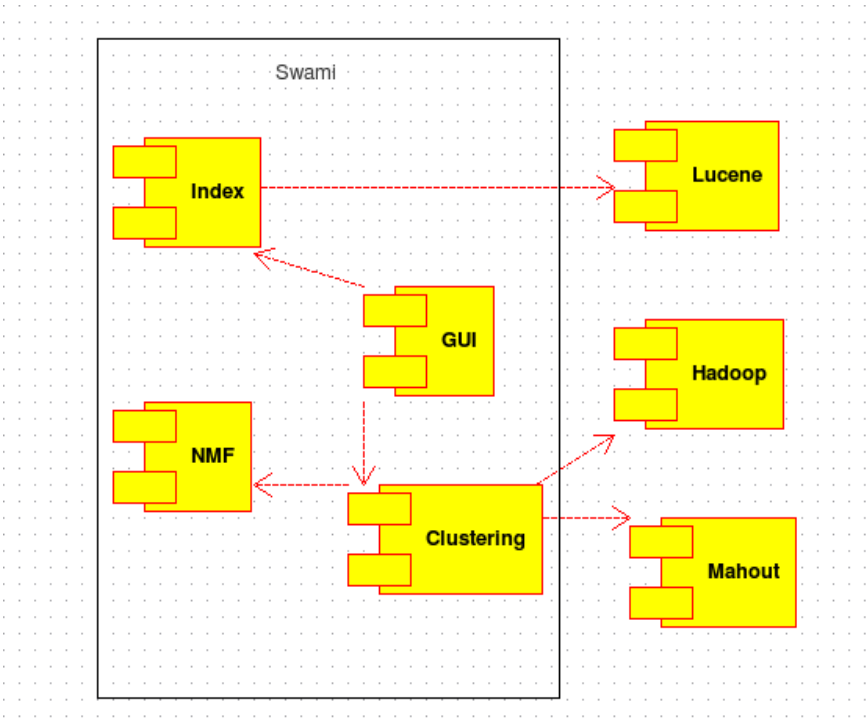


Figure 4.2: The Component Diagram

The major components of Swami are show in the figure above. The core components are shown inside the box and the dependent components are shown outside. The index component has classes used for reading/writing to index, the Lucene Document and a classes to create document-term matrix from the index. The NMF component has classes for the NMF algorithm by Lee and Seung [3] and utility class to help perform operation like extracting the top words in features, top files in features etc. Clustering component has classes to help run clustering and finally the GUI component has the class for user interface and the main class of the application. External components are Lucene that has been used for indexing, the Mahout APIs for running map-reduce operation and Hadoop to start the Hadoop Distributed File System.

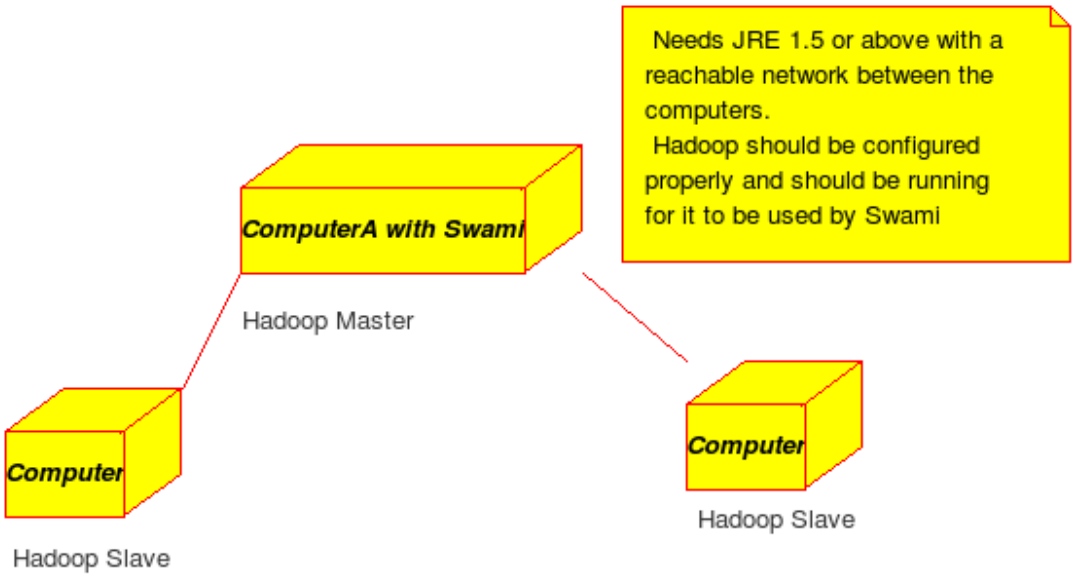


Figure 4.3: The Deployment Diagram

For deployment of Swami since Hadoop isn't a must, it can be easily started in a single computer. Even using the pseudo-distributed mode of Hadoop doesn't require multi-node. While running Hadoop in fully-distributed mode multiple nodes are required which can be represented in the above deployment diagram.

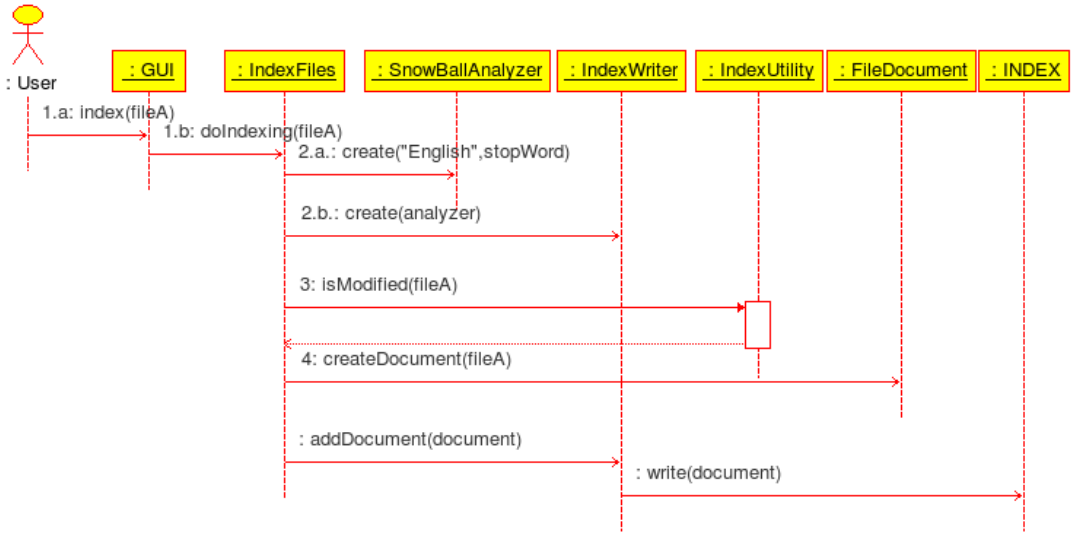


Figure 4.4: The Sequence Diagram for Indexing

Before the documents can be clustered, they need to be indexed. For the purpose of indexing, Lucene APIs have been utilized. The documents are determined whether they are up-to-date in index. If it's up to date then do nothing what follows. If the document is new, create a Lucene Document, if it's not updated then delete the old document and create new Lucene Document. The stop-words are removed using key-words from [6] and Potter Stemming. [4] is applied. The stray files are removed which is not shown in the figure. (see section 3.1.2)

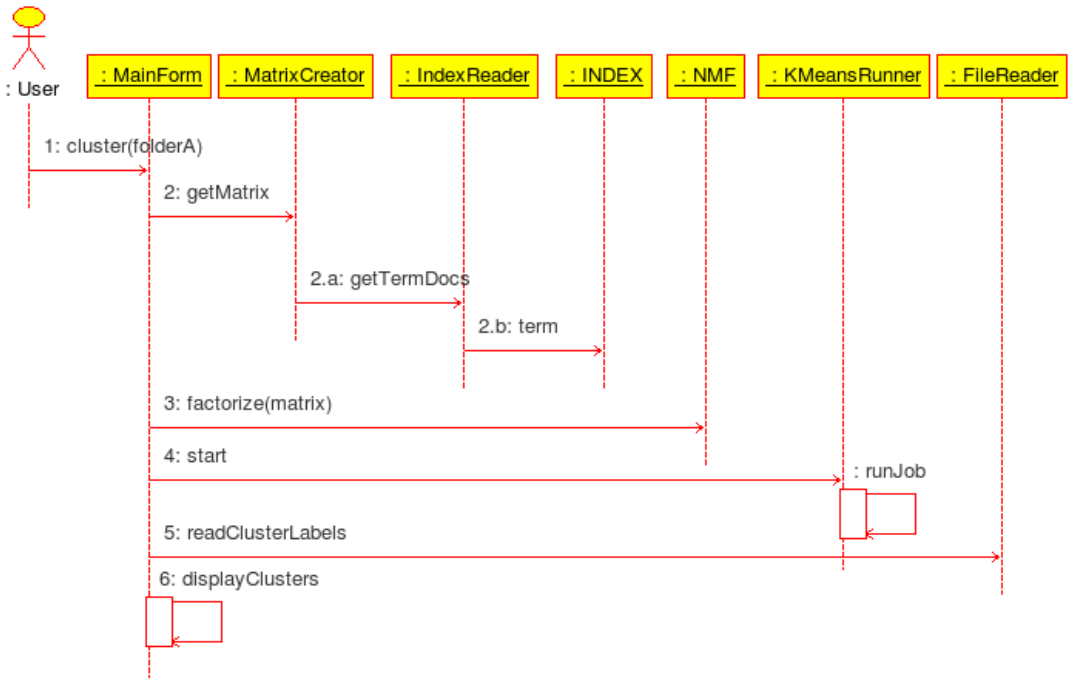


Figure 4.5: The Sequence Diagram for Clustering

Clustering has more complex steps than Indexing the documents. It involves creation of document-term matrix. Then users can choose to use NMF for clustering. They may or may not use Hadoop as well. The final clusters are read from file if Hadoop is used for clustering. Somewhat closer picture is described in the above diagram.

4.3 Interfaces

Figure 4.6: Indexing Files/Folders

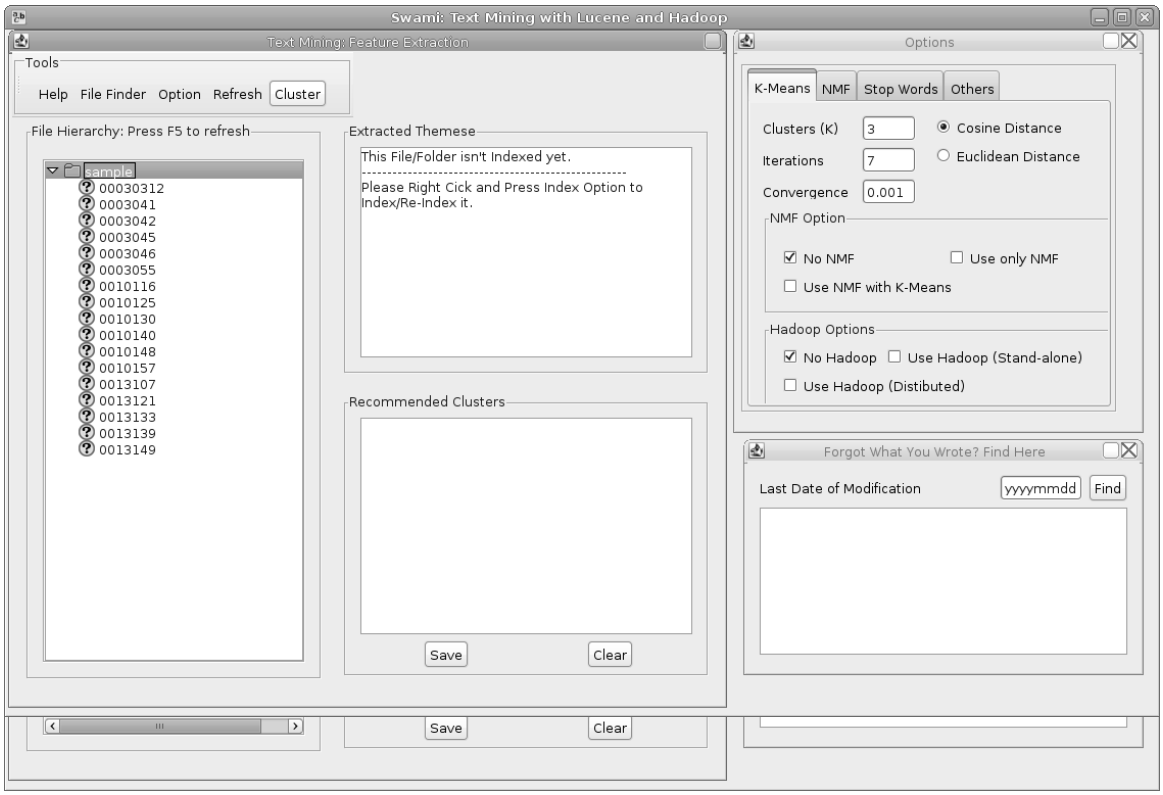


Figure 4.7: Clustering

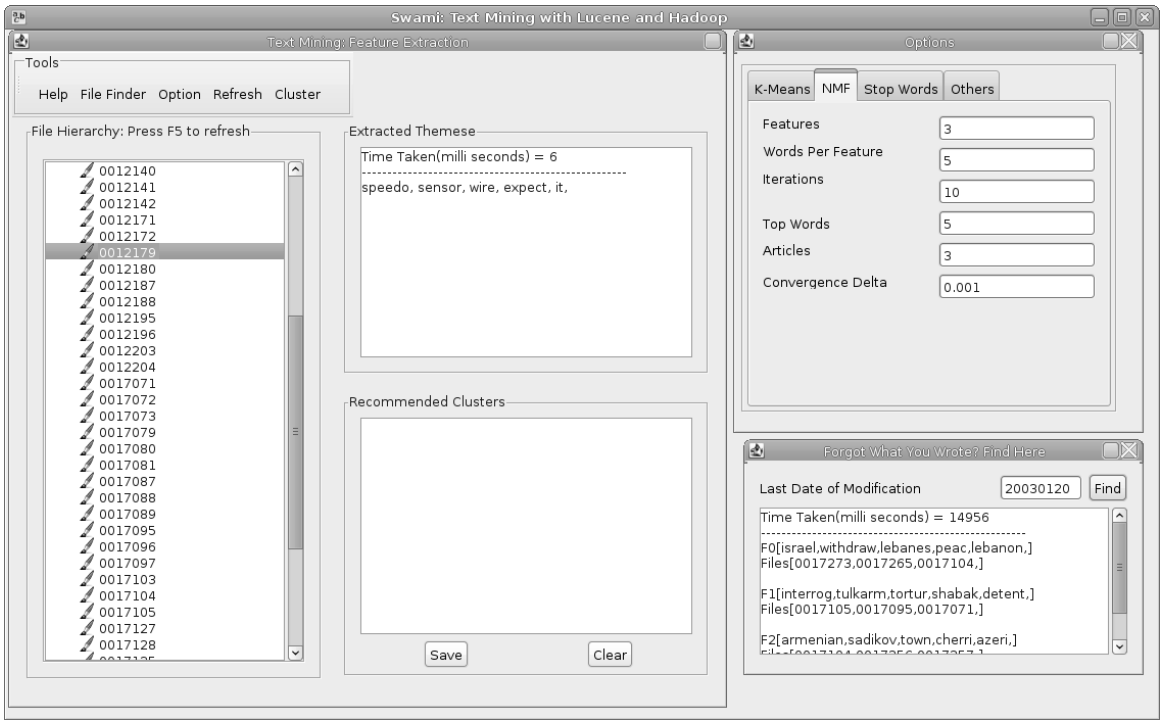
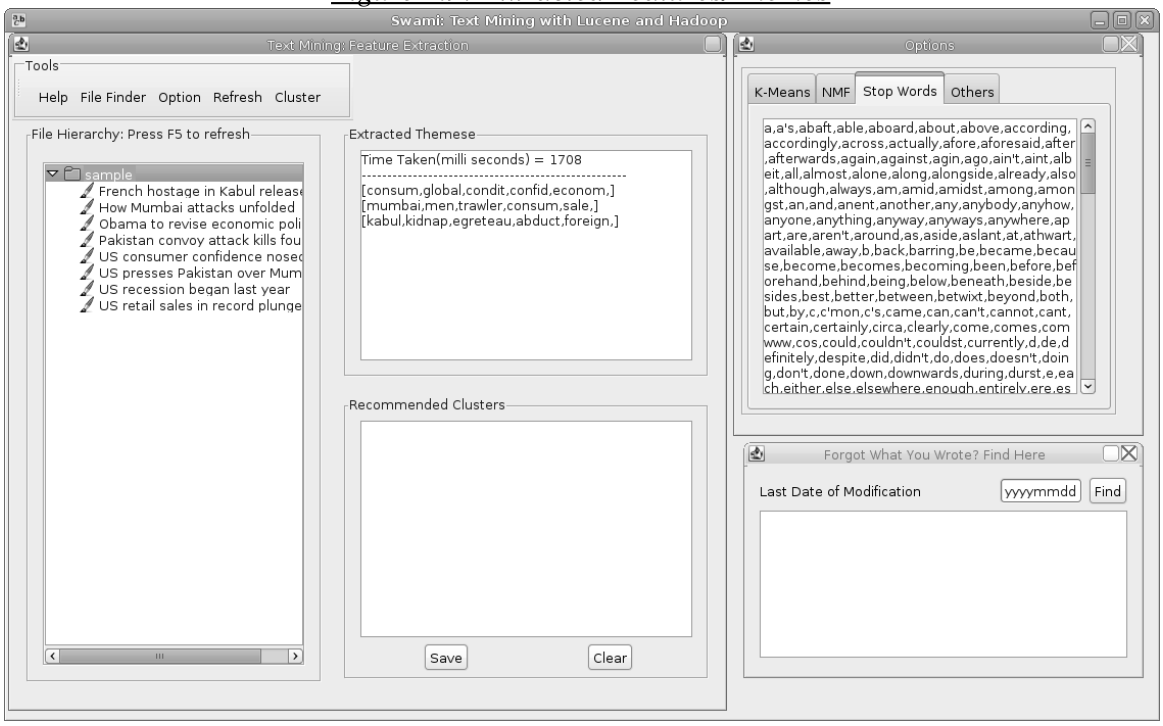


Figure 4.8: File finding

Figure 4.9: Extracted Features/Themes



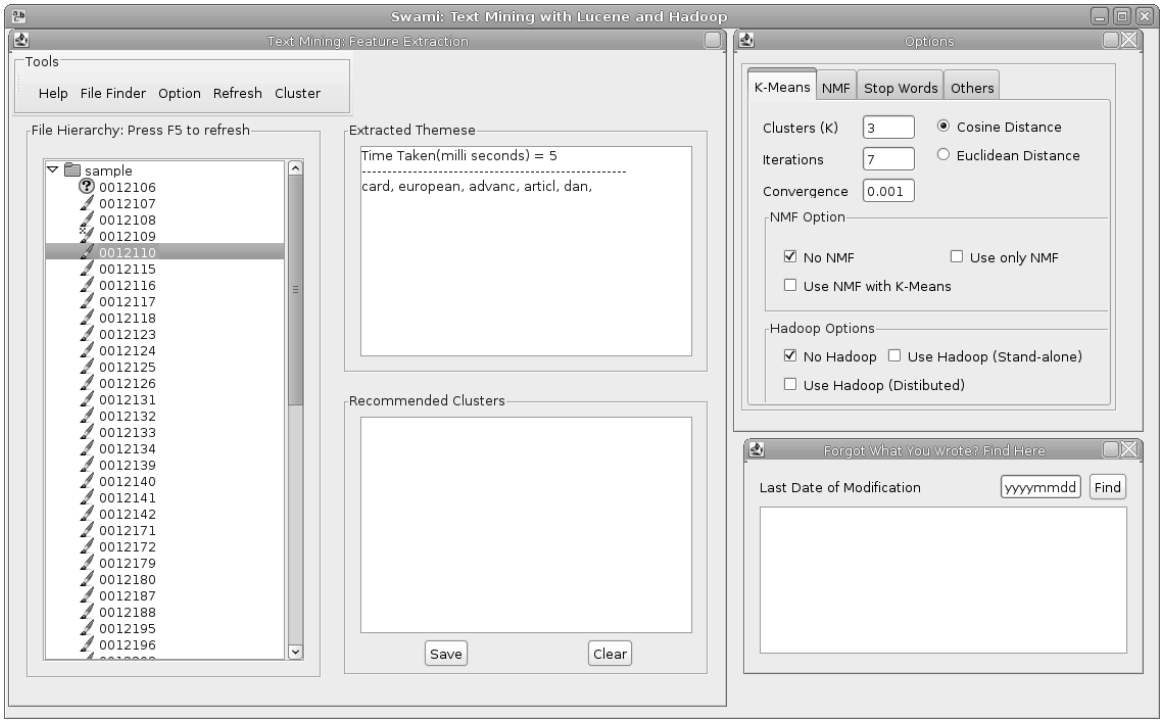


Figure 4.10: Top Words

CHAPTER 5

EXPERIMENTS AND RESULTS

To work with this model, the standard text data set 20 News Groups[4] was used for document clustering with NMF. Aforementioned application, *Swami* was used for the purpose of experimentation. This section describes the data set used, experimental parameters and the results.

5.1 Data set Description

20 News Groups is quite a popular data set for text clustering and classification. It has a collection about 20,000 documents across 20 different newsgroups from Usenet. Each newsgroup is stored in a subdirectory, with each article stored as a separate file. Some of the newsgroups are closely related with each other while some are highly unrelated. Below is the topics of the newsgroups arranged by Jason Renn[5]

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

Table 5.1: List of Topics of 20 New Groups

4 http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html

5 http://people.csail.mit.edu/jrennie/

5.2 Experiment

For the purpose of experimentation, clustering was done using up to 10 group. 5 documents were taken randomly for two group each and added to a folder. The folder was indexed after removing the stop-words using KEA stop-words [6] and applying Porter stemming[4]. Then the clustering was done and results were noted. Next 5 documents were taken out randomly from another group, added to the folder, indexed and clustering was done accordingly. In this way a total of 10 groups with 50 documents were clustered. Clustering results were noted for three cases, without using Hadoop, using Hadoop in local mode and finally using Hadoop in pseudo-distributed mode.

For KNMF the following parameters were used

1. NMF: convergence parameter = 0.001 and maximum iteration = 10.
2. K-Means: k = number of news groups in folder, convergence parameter = 0.001, maximum iteration = 1, distance measure = cosine

Since the length of W was not normalized as suggested by Xu et al. [9] there was no unique solution. For this purpose the experiments the highest values of AC among the three cases as mentioned above was noted.

The performance of the clustering algorithm is evaluated by calculating the accuracy defined in [9] as follows:

Given a document d_i , let l_i and α_i be the cluster label and the label provided by the document corpus, respectively. The AC is defined as follows:

$$AC = \frac{\sum \delta\left(\alpha_i, map\left(l_i\right)\right)}{n} \qquad (10)$$

where n denotes the total number of documents, $\delta(x, y)$ is the delta function that equals one if x = y and equals zero otherwise, and $map(l_i)$ is the mapping function that maps each cluster label l_i to the equivalent label from the document corpus.

5.3 Results

Table below shows the time taken by KNMF algorithm on the 20 Newsgroup collection on a Linux Ubuntu 8.04 laptop (1.66Ghz Intel Pentium Dual-core, 1G RAM) with and without MapReduce (Map = 2). The number of clusters were denoted by k, AC denotes the accuracy measure. The without Hadoop and Local Reference mode of Haoop shows time taken by KNMF as *time by NMF/the total time taken*. The pseudo-distributed mode of Hadoop shows time for Map phases.

k	**AC**	**Without Hadoop**	**Local Reference mode of Hadoop**	**Pseudo-Distributed mode of Hadoop**
2	0.80	0.558/0.597	0.390/1.580	1/2
3	0.75	0.898/0.958	0.796/2.0	1/2
4	0.66	1.090/1.159	1.074/2.307	2/2
5	0.60	1.961/2.111	2.155/3.457	2/2
6	0.56	4.086/5.295	4.122/5.617	1/1
7	0.68	6.340/7.158	6.262/7.653	2/1
8	0.625	8.710/9.874	8.615/10.025	2/2
9	0.533	12.435/14.088	12.503/14.027	3/3
10	0.60	26.963/30.615	26.700/30.648	3/3

Table 5.2: Results

The chart below the time taken for performing only the clustering phase. Time taken for clustering phase is calculated from the above table as (*the total time taken - time by NMF)*. For the pseudo-distributed mode of Hadoop, the time taken by map phase is considered time taken for clustering. It can be seen that the time by the map phase of pseudo-distributed mode of Hadoop is quite steady and rises only when the number of clusters increase to 8. The time taken by local refrence and serial implementation or without using Hadoop exceeds time taken by pseudo-distributed mode of Hadoop for cluster size equivalent to 10.

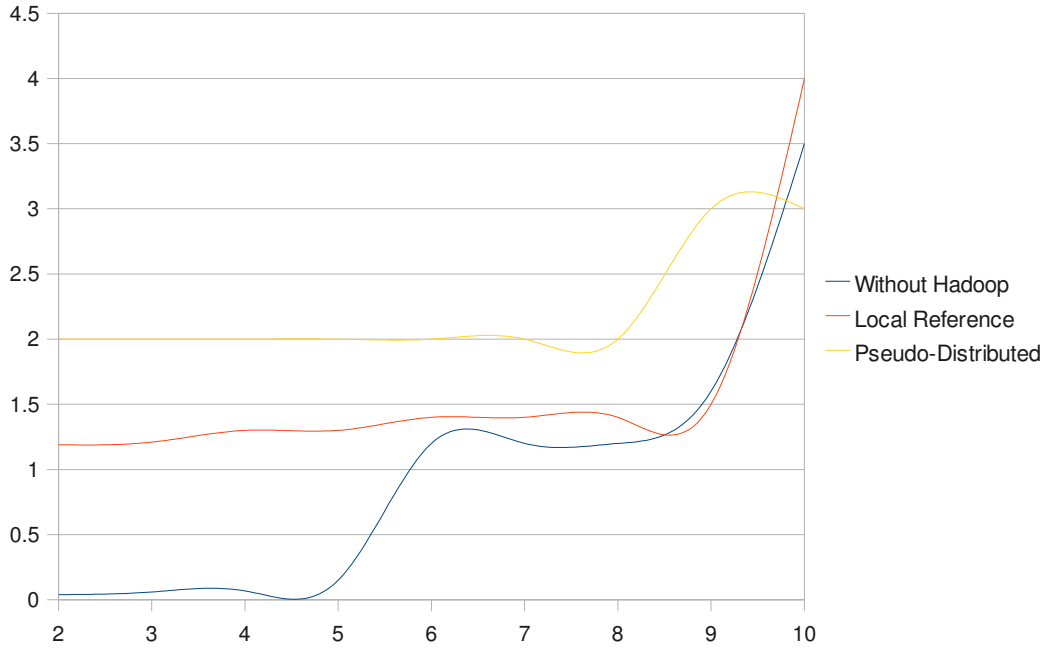


Figure 5.1: Time taken by the clustering phase (k-means with 1 turn)

CHAPTER 6

CONCLUSION

In this work, a new working model for document clustering was given in this work along with development of application based on this model. This application can be used to organise documents into sub-folders without having to know about the contents of the document. This really improves the performance of information retrieval in any scenario. The accuracy of model was tested and found to be 80% for 2 clusters of documents and 75% for 3 clusters and the results averages to 65% when for 2 through 10 clusters. NMF has shown to be a good measure for clustering document and this work has also shown similar results when the extracted features are used as the final cluster labels for k-means algorithm. To scale the document clustering the proposed model uses the map-reduce implementation of k-means from Apache Hadoop Project and it has shown to scale even in a single cluster computer when clusters size exceeded 9 i.e. 40 documents averaging 1.5 kilobytes.

CHAPTER 7

FUTURE WORKS

This research has tried to introduce a new working model to document clustering. A detail experimentation on various corpus with the proposed method will be very helpful for this work. Study of performance in fully distributed mode of Hadoop with large number of clusters/processors could shed more light on the scalability. Inclusion of auto detection of number of topics [18] can be quite useful to the built software. Since the software build on the purposed model has its limitation to index only the .doc version and text files, indexing documents in various formats with open source frameworks like apache Tika can be continued from this work. Furthermore, the parallel implementation of NMF by Kanjani K in [12] and Robila et al. [13] and indexing could be a good extension for the software.

REFERENCES

[1] Cutting, D, Karger, D, Pederson, J & Tukey, J (1992). Scatter/gather: A cluster-based approach to browsing large document collections. In Proceedings of ACM SIGIR.

[2] Lee, D & Seung, H (1999). Learning the Parts of Objects by Non-negative matrix factorization in *Nature* 401, 788–791.

[3] Lee, D & Seung, H (2001). Algorithms for non-negatvie matrix factorization. In T. G. Dietterich and V. Tresp, editors, Advances in Neural Information Processing Systems, volume 13. Proceedings of the 2000 Conference: 556-562,*The MIT Press*.

[4] Porter, MF (1980). "An algorithm for suffix stripping", *Program*, Vol. 14, No. 3, pages 130-137
http://tartarus.org/~martin/PorterStemmer/def.txt

[5] The Porter Stemmeing Algorithm in Snowball
http://snowball.tartarus.org/algorithms/porter/stemmer.html

[6] Key Phrase Extraction Algorithm (KEA)
http://www.nzdl.org/Kea/

[7] Teknomo, K. K-Means Clustering Tutorials.
http://people.revoledu.com\kardi\tutorial\kMean\

[8] Guduru, N (2006). Text mining with support vector machines and non-negative matrix factorization algorithm. *Masters Thesis. University of Rhode Island, CS Dept.*

[9] Xu, W, Liu, X & Gong, Y (2003). Document clustering based on non-negative matrix factorization. Proceedings of ACM SIGIR, pages 267–273.

[10] Yang, CF, Ye, M & Zhao, J (2005). Document clustering based on non-negative

sparse matrix factorization. International Conference on advances in Natural Computation, pages 557–563.

[11] Ding, C, He X, & Simon, HD (2005). On the Equivalence of Nonnegative Matrix Factorization and Spectral Clustering. Proceedings in SIAM Intternational Conference on Data Mining, pages 606-610.

[12] Kanjani, K (2007). Parallel Non Negative Matrix Factorization for Document Clustering.

[13] Robila, SA & Maciak, LG (2006). A parallel unmixing algorithm for hyperspectral images. Technical report, Center for Imaging and Optics, Montclair State University,.

[14] Strehl, A, Ghosh, J & Mooney, RJ (July 2000), "Impact of similarity measures on web-page clustering," in AAAI Workshop on AI for Web Search, pages 58-64.

[15] Hoyer, P (2002). Non-Negative Sparse Coding. In Proceedings of the IEEE Workshop on Neural Networks for Signal Processing, Martigny, Switzerland.

[16] Pauca, V, Shahnaz, F, Berry, MW & Plemmons R (April 22-24, 2004). Text Mining Using Non-Negative Matrix Factorizations. In Proceedings of the Fourth SIAM International Conference on Data Mining, Lake Buena Vista, FL.

[17] Amy, L & Carl, M (2006). ALS Algorithms Nonnegative Matrix Factorization Text Mining.

[18] Guillamet, D & Vitria, J (2002). Determining a Suitable Metric when Using Non-Negative Matrix Factorization. In Sixteenth International Conference on Pattern Recognition (ICPR'02), Vol. 2, Quebec City, QC, Canada.

[19] Dhillon, SI & Modha, DS (2001). Concept decompositions for large sparse text data using clustering.

[20] Berry, M, Dumais, ST & O'Brien, GW(1995). Using Linear Algebra for Intelligent Information Retrieval. Illustration of the application of LSA to document retrieval.

[21] Landauer, T, Foltz, PW & Laham, D(1998). Introduction to Latent Semantic Analysis.. Discourse Processes 25: pages 259–284

[22] Drost, I (November 2008). Apache Mahout : Bringing Machine Learning to Industrial Strength. In Proceedings of ApacheCon 2008, pages 14-29, New Orleans

[23] Michels, S (July 5, 2007). Problem Solving on Large-Scale Clusters, Lecture 4.

[24] Gillick, D, Faria, A & DeNero, J (December 18, 2006). MapReduce: Distributed Computing for Machine Learning.

[25] Dean, J & Ghemawat, J (December 2004). MapReduce: Simplified Data Processing on Large Clusters. In the Proceedings of the 6th Symp. on Operating Systems Design and Implementation.

[26] Chu, CT, Kim, SK, Lin, YA, Yu, YY, Bradski, G, Yng, Andrew, & Olukotun, K (2006). Map-Reduce for Machine Learning on Multicore, *NIPS*

[27] Ranger, C, Raghuraman, R, Penmetsa, A, Bradski, G & Kozyrakis, C (2007). Evaluating MapReduce for Multi-core and Multiprocessor Systems