Document Clustering Through Non-Negative Matrix Factorization: A Case Study of Hadoop for Computational Time Reduction of Large Scale Documents.

Bishnu Prasad Gautam and Dipesh Shrestha

● Abstract

In this paper we discuss a new model for document clustering which has been adapted using non-negative matrix factorization during our research. 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 mapreduce framework of apache hadoop project was used for parallel implementation of kmeans algorithm from apache mahout project. Since experiments were carried only in one cluster of Hadoop, the significant reduction in time was obtained by mapreduce 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 kmeans, and for large scale documents the parallel implementation using mapreduce can lead to reduction of computational time.

● Key words

Document Clustering
Non-negative Matrix
Map Reduce

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 is 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 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.

1.2 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 describe the documents. The extraction of these features from the documents in called 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 [5].

1.3 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. In order to overcome the shortcomings of VSM model, LSI estimate the structure in word usage through truncated Singular Value Decomposition (SVD). Retrieval is then performed using a database of singular values and vectors obtained from the truncated SVD. Application of Latent Semantic Indexing with results can be found in Berry et al. [14] and Landauer et al in [15].

1.4 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 Vmn into the product of two lower rank matrices Wmk and Hkn, such that Vmn is approximately equal to W_{mk} times H_{kn} .

$$V_{mn} \approx W_{mk} \cdot H_{kn} \tag{1}$$

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 [13]. 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 vi 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,

$$V_i \approx W_1 \cdot h_{i1} + W_2 \cdot h_{i2} + ... + W_k \cdot h_{ik} \tag{2}$$

NMF uses an iterative procedure to modify the initial values of Wmk and Hkn so that the product approaches Vmn . 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 [2], sparse encoding by Hoyer [10], gradient descent with constrained least squares by Pauca [11] and alternating least squares algorithm by Pattero [12]. 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

$$\min_{W,H} \|V - WH\|_F^2 \quad (3)$$

where W and H are non-negative. The method proposed by Lee and Sung [2] based on multiplicative update rules using Forbenus norm, popularly called multiplicative method (MM) can be described as follows.

1.4.1 MM Algorithm

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

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

(a) $$H_{cj} \leftarrow H_{cj} \frac{\left(W^T V\right)_{cj}}{\left(W^T WH\right)_{cj} + e} \quad (4)$$

(b) $$W_{ic} \leftarrow W_{ic} \frac{\left(VH^T\right)_{ic}}{\left(WHH^T\right)_{ic} + e} \quad (5)$$

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 [2] proved that with the above update rules objective function (1) 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.

1.5 Document clustering with NMF

Ding C et. al in [8] 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 [6] where unit euclidean distance constraint was added to column vectors in W. Yang et al. [7] 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. "[6]

In the above statement, $W \approx UV$

From the work of Kanjani K [9] it is seen that the accuracy of algorithm from Lee and Seung [2] is higher than their derivatives [9,10]. In this work, the original multiplicative update proposed by Lee and Seung in [2] is undertaken.

2. 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 contain explanation brief introduction to the underlying architecture of Hadoop Distributed File System (HDFS) used to run k-means algorithm.

2.1 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 $F=\{f_1, f_2, f_3 \ldots f_k\}$ computed by NMF. Let the documents in the term-document matrix be $V = \{d_1, d_2, d_3 \ldots d_n\}$, then document di is said to belong to cluster fx if, the angle between di and f_x is minimum.

2.1.1 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 [2] on V and get W and H using (1)
4. Apply cosine similarity to measure distance between the documents di and extracted features/vectors of W. Assign di to wx if the the angle between di 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.

2.1.2 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 [4] 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 [3].

2.2 Parallel implementation of k-means

The parallel implementation strategy of k-means algorithms in multi-core is described in [20] as:

"In k-means [9], 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 project1 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 [17]. Drost, I [16] describes k-means of Mahout project. In [18] Gillick et. Al studied the performance of Hadoop's implementation of MapReduce and has suggested performance enhancing guidance as well.

2.2.1 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 centred 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 $V = \{v_1, v_2, ...vn\}$ and list of feature vectors $W= \{w_1, w_2, w_3\}$ computed by NMF. Then,

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

where w_x is the closest (cosine similarity) feature vector to data vector v_i.

2.3 Hadoop

Hadoop is a distributed file system written in Java with an additional implementation of Google's MapReduce framework [19] 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)

2.3.1 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 determines 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, 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 Local(stand-alone), Pseudo-distributed mode or Fully-distributed mode.

2.3.2 MapReduce framework in Hadoop3

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 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 schedulin4g 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.

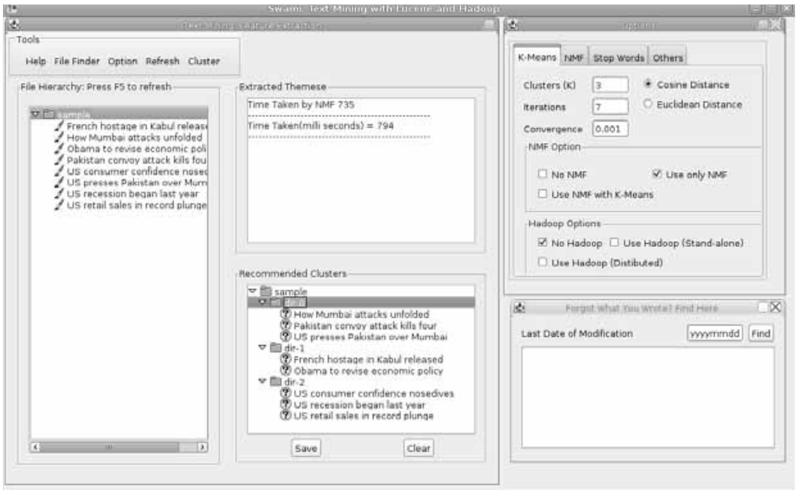
3. Implementation

Since Hadoop, Lucene and Mahout are built with Java natively, it would be easy for the interoperability between the components developed with Java. Considering this fact, Java was chosen as the programming language for the implementation of our proposed model.

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, a Lucene Document is created, if it's not updated then old document are deleted and new Lucene Document is created. The stop-words are removed using key-words from [4] and Potter Stemming. [3] is applied. Section 2.1.2 describes the Indexing in details.

Clustering has more complex steps than Indexing the documents. It involves creation of document-term matrix followed by the application of KNMF. Section 2.1.1 describes the steps in detail.

Figure 3.1: Clustering with our model



4. Experiments and Results

4.1 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 are the topics of the newsgroups arranged by Jason Renn[5]

Table 4.1: List of Topics of 20 New Groups

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

4.2 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 [4] and applying Porter stemming[3]. 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. [6] 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 [6] as follows:

Given a document di, let li and αi be the cluster label and the label provided by the document corpus, respectively. The AC is defined as follows:

$$AC = \frac{\sum \delta(\alpha_i, map(l_i))}{n} \qquad (6)$$

where n denotes the total number of documents, δ(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.

4.3 Results

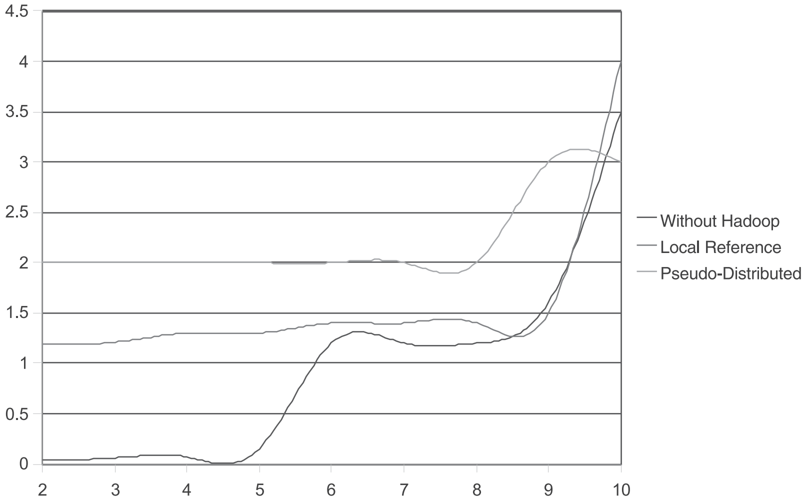
Table below shows the time taken by KNMF algorithm on the 20 Newsgroup collection on a Linux Ubuntu 8.04 laptop (1.66Ghz Intel Pentium Dual-core, 1G RAM) with and without MapReduce (Map = 2). The numbers of clusters were denoted by k, AC denotes the accuracy measure. The without Hadoop and Local Reference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

Table 4.2: Results

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

The chart below shows 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 reference and serial implementation or without using Hadoop exceeds time taken by pseudo-distributed mode of Hadoop for cluster size equivalent to 10.

Figure 4.1: Time taken by the clustering phase (k-means with 1 turn)



5. Conclusion

In this work, a new working model for document clustering was given 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.

To enhance the scalability of our model, it is required to test the system by using fully-distributed mode of Hadoop. Furthermore, we have tested the only the text and .doc format of files for indexing, our future works will be concentrated to support the other formats of the documents.

● 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 (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.*

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

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

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

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

[7] 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.

[8] 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.

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

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

[11] 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.

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

[13] 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.

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

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

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

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

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

[19] 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.

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

Footnotes:

1. http://lucene.apache.org/mahout/
2. http://hadoop.apache.org/core/
3. http://hadoop.apache.org/core/docs/current/mapred_tutorial.html
4. http://people.csail.mit.edu/jrennie/