CLUSTERED DISTRIBUTED INDEX FOR EFFICIENT TEXT RETRIEVAL USING THREADS

M. Basavaraju [1] and Dr. R. Prabhakar [2]

[1]Research Scholar, Dept. of CSE, Coimbatore Institute of Tech., Anna University, Coimbatore, Tamil Nadu, India,
Professor & Head, Dept. of CSE, Atria Institute of Tech., Bengaluru, Karnataka, India,
Email : basavaraju@atria.edu
[2] Professor-Emeritus Dept. of CSE, Coimbatore Institute of Tech., Coimbatore, Tamilnadu, INDIA

ABSTRACT

In this research paper, a novel method of improving the clustered distributed indices for efficient text retrieval using threads is presented. In text retrieval, text search refers to a technique of searching stored document or database. In a full text search, the search engine examines all the words in every stored document as it tries to match search words supplied by the user. When dealing with a small number of documents, the full-text search engine performs a serial scan, where it directly scans the contents of the documents with each query. When the number of documents to search is potentially large or the quantity of search queries to perform is substantial, the problem of full text search is often divided into two tasks, viz., indexing and searching. The indexing stage scans for text of all the documents and builds a list of search terms, often called an index. In the search stage, when performing a specific query, only the index is referenced rather than the text of the original documents. Considering all the above mentioned criterias, this paper aims at improving the search time on the index, by clustering the index. Threads are used to perform a parallel search on each of these clusters. The algorithm developed in C has been tested on various sizes of data and queries and compared with the sequential search method. The depicted results shown in the result section clearly show that this approach improves the search time significantly & the method proposed shows the efficacy, effectiveness, which can be further implemented for real time applications.

KEYWORDS

Clustering, Distributed index, Threads, Text retrieval, Posting list, Query processing, Algorithms, Performance.

1. INTRODUCTION

Recent times have seen an explosive growth in the availability of various kinds of data. It has resulted in an unprecedented opportunity to develop automated data-driven techniques of extracting useful knowledge. This has led to the concept of data mining. Data mining is an important step in this process of knowledge discovery & consists of methods that discover interesting non-trivial & useful patterns hidden in the data [21]. Vast amounts of unstructured documents are available in many fields, and when we consider the web, documents can be retrieved from all over the world. A large amount of text has to be shifted to retrieve the information useful to the user. Fast and effective clustering is a fundamental tool in unsupervised learning. Various access methods have been developed to support efficient search

DOI : 10.5121/ijgca.2010.1201

and retrieval over text document collections. Examples include suffix arrays, inverted files [22] or inverted indexes [23], Witten et. al. and signature files [24].

A rough but widely agreed upon framework is to classify clustering techniques as Hierarchical Clustering and partitional clustering, based on the properties of the generated clusters [3]. The distance metric (Euclidean, Hamming, etc) must be appropriately chosen according to the underlying shapes of the data, whether it is spherical or ellipsoidal data [2]. Queries can be either boolean or ranked. A boolean query is made up of terms connected by the logical operators AND, OR, NAND and NOT-can be used to identify the documents containing a given combination of terms and is similar to the kind of query used on relational tables [2].

Ranking, on the other hand, is a process of matching an informal query to the documents and allocating scores to documents according to their degree of similarity to the query [1]. The major problem is to organize and store huge amounts of data. When search is done on this data the quality of the search should be very good. The search should also be cost effective and time effective. It is very important that the result obtained by the search query should be equivalent to what we are actually searching for [4].

Foti et.al. developed scalable parallel clustering models for data mining on multi-computers in their research paper in [16]. They designed & implemented on MIMD parallel machines of P-AutoClass, a parallel version of the AutoClass system based upon the Bayesian method for determining optimal classes in large datasets. In particular, efficiency and scalability of P-Autoclass vs. the sequential Autoclass system were also evaluated and compared by them.

Text search and information retrieval in the modern day computing plays a very important role. This can be done using the highly efficient search engines such as google, yahoo, rediff & several others. Search engines are tools for finding the documents in a collection that are good matches to user queries [5]. Typical kinds of document collection include web pages, newspaper articles, academic publications, company reports, research grant applications, manual pages, encyclopaedias, parliamentary proceedings, bibliographies, historical records, electronic mail, and court transcripts. These collections range dramatically in size [19].

The plain text of a complete set of papers written by a researcher over ten years might occupy 10 megabytes, and the same researcher's (plain text, non-spam) 10-year email archive might occupy 100 megabytes [6]. A thousand times bigger, the text of all the books held in a small university library might occupy around 100 gigabytes. In 2005, the complete text of the Web was probably some several tens of terabytes. Collections also vary in the way they change over time. A newswire archive or digital library might grow only slowly, perhaps by a few thousand documents a day; deletions are rare [19].

Web collections, in contrast, can be highly dynamic. Fortunately, many of the same search and storage techniques are useful for these collections. Text is not the only kind of content that is stored in document collections [7]. Research papers and newspaper articles include images, email includes attachments, and web collections include audio and video formats. The sizes discussed previously are for text only; the indexing of media other than text is beyond the scope of this tutorial [19].

Several methods have been proposed by various researchers in their paper in order to solve this problem. One of the methods could be ranking. Ranking is a process of matching a query to the documents and allocating scores to the documents according to their degree of similarity [2]. The other method is the process of clustering. Clustering is a process of organizing the objects into groups, whose members are similar in some way.

Search engines are structurally similar to database systems. Documents are stored in a repository, and an index is maintained. Queries are evaluated by processing the index to identify matches which are then returned to the user [8]. However, there are also many differences. Database systems must contend with arbitrarily complex queries, whereas the vast majority of queries to search engines are lists of terms and phrases. In a database system, a match is a record that meets a specified logical condition; in a search engine, a match is a document that is appropriate to the query according to statistical heuristics and may not even contain all of the query terms [19].

Database systems return all matching records; search engines return a fixed number of matches, which are *ranked* by their statistical similarity. Database systems assign a unique access key to each record and allow searching on that key; for querying on a web collection, there may be many millions of documents with nonzero similarity to a query [9]. Thus, while search engines do not have the costs associated with operations such as relational join, there are significant obstacles to fast response, that is, a query term may occur in a large number of the documents, and each document typically contains a large number of terms. The challenges presented by text search have led to the development of a wide range of algorithms and data structures. These include representations for text indexes, index construction techniques, and algorithms for evaluation of text queries [19].

The paper is organized in the following sequence. Section 2 presents an overview of the related work done. Section 3 presents the concept of indexing, followed by the threading concepts in section 4. The section 5 depicts the process of clustering. The leaders algorithm developed to obtain the search indices is presented in section 6. Searching process is dealt with in the section 7. The simulation results along with the detailed discussions are presented in the section 8. This is followed by the conclusions in the section 9 & the scope for future works.

2. RELATED WORK

Indexes based on these techniques are crucial to the rapid response provided by the major web search engines. Through the use of compression and careful organization, the space needed for indexes and the time and disk traffic required during query evaluation are reduced to a small fraction of previous requirements [10]. Thus, considering all the above mentioned parameters, we have tried to develop a novel method of improving the clustered distributed indices by developing a code in C language, which is the highlight of this research paper.

We are using Leader's algorithm to cluster the huge data that we have obtained [3]. When we are searching, we will not have to search the entire data, instead we would have to search only the cluster. This is the greatest advantage of using clustering. By clustering, the search becomes cost effective and time effective. Hence, we have used clustering in our search techniques. The input to our search engine is the data which is collected from several books. We have collected the table of contents of several books in different areas. We first compress the raw data that we have obtained by using methods such as stemming and stopping [11].

Stopping takes in the various tables of content as input, processes them and gives the output which are free of stop words. Stop words are those which include repetitive words or functional words like 'and', 'or', 'but', 'because', etc., which are not of much importance at the time of storage in the database but may just use up memory [12]. Hence, these words are being eliminated or cleaned up before processing it further for the later stages. Case folding is also done within this stage where the uniformity of text in lower case is been done.

Ex. : Let us consider an example 1 ….

Input : Hello, Welcome to this Atria College

Output : hello welcome atria college

Now, let us look at the above example. The input statement is fed into the stop module. The output that we obtain is free from punctuations and the commonly occurring words are removed [13]. The next phase takes in stop free words list as the input. The job of stemming is to remove the different forms of the same word and just retain the base word. For ex., 'writing', 'writes', 'written' ….. can all be reduced to 'write' and only 'write' gets stored in the vocabulary or database. Hence the base word gets stored and other forms of that word are being cleaned off, thus saving the memory space & also increasing the efficacy of the search techniques. These steps of stopping, stemming and case folding are all part of pre-processing stages where the initial data is being cleaned before actually processing it [14].

For dealing efficiently with high traffic of user queries, we use distributed inverted indices. An inverted file is composed of a vocabulary table and a set of posting lists. The vocabulary table contains the set of relevant terms found in the test collection which is arranged in the alphabetical order. Each of these terms is associated with a posting list which contains the document identifiers along with additional data used for ranking purpose. To solve a query, it is necessary to get the set of documents associated with the query terms and then perform a ranking of these documents in order to select the top '*K*' documents as query answer [15].

Depending on the hamming distance between the words in the vocabulary, words which are similar such as 'write' are clustered. Several clusters exist depending on the vocabulary size and the kind of documents. Threads (light weight process) are applied to individual clusters. Surprisingly our results show that by using the concept of clustering and parallelism, search becomes more cost effective, time effective and the quality of the search becomes accurate. Our results show that this strategy is able to cause efficient performance both in large scale and small scale search engines [16].

3. INDEXING

In order to avoid linear scanning of the text corpus for each query, we index the documents well in advance. To obtain the speed benefits of indexing, we also need to perform the indexing of the documents well in advance [1]. For doing this operation, the major steps required are as follows :

- Collect the documents to be indexed.
- Tokenize the text, breaking documents into tokens.
- Produce a list of indexing terms.
- Create an inverted index consisting of a list and a vocabulary

Ex. : Let us consider another example 2 ….

DOC1 : apple baby cream
DOC2 : pineapple cream jam
DOC3 : mango pineapple

Term	Doc Id
Apple	1
Baby	1
Cream	1, 2
Jam	3
Mango	3
Pineapple	2

Matrix representation: This is the representation of the term versus the various documents. We could compute the distance between the two words by using some distance metric such as a Euclidean Distance or a Hamming Distance as computed below [17].

Term	DOC1	DOC2	DOC3
Apple	1	0	0
Baby	1	0	0
Cream	1	1	0
Jam	0	0	0
Mango	0	0	1
Pineapple	0	1	1

Term document incidence matrix : An index always maps back from terms to the parts of a document where they occur. We keep a dictionary of terms which is usually sorted in the alphabetical order. For each term, we have a list that records which documents the term occurs in [18]. Each item in the list records that a term appeared in a document is called "*Posting*". The list is called '*Posting List*'.

The dictionary is commonly kept in the memory, while posting lists are normally kept on the disk. For an in-memory posting list, we have used a singly linked list. The posting list contains the following fields : The first field indicates the frequency of the occurrences of the term in the document. The second field indicates the document identity (Doc id) and third field points to the next list.

4. THREADING PROCESS

A thread is a light weight process (LPW). A thread of execution results from a fork of a computer program or more concurrently running tasks. The implementation of threads and processes differs from one operating system to another, but in most cases, a thread is contained inside a process. Multiple threads can exist within the same process and share resources such as memory, while different processes do not share these resources [19].

On a single processor, multi-threading generally occurs by time-division multiplexing (as in multi-tasking) & the processor switches between different threads. This context switching generally happens frequently when the user perceives the threads or tasks when running at the same time. On a multi-processor or multi-core system, the threads or tasks will generally run at the same time, with each processor or core running a particular thread or task. Generally, the support for threads in programming languages varies. A number of languages simply do not support having more than one execution context inside the same program executing at the same time [20].

Threads differ from traditional multitasking operating system processes in that:

- processes are typically independent, while threads exist as subsets of a process,
- processes carry considerable state information, where multiple threads within a process share state as well as memory and other resources,
- processes have separate address spaces, where threads share their address space,
- Processes interact only through system-provided inter-process communication mechanisms.

Context switching between threads in the same process is typically faster than context switching between processes. In the following paragraph, information about the pthreads is provided. '*Pthreads*' specifies a thread's starting point as a procedure name, other thread packages differ in their specification of even this most elementary of concepts. Pthreads is a standardized model for dividing a program into sub-tasks whose execution can be interleaved or run in parallel. The '*P*' in Pthreads comes from 'POSIX' (Portable Operating System Interface). Pthreads is a defined set of C language programming types and calls with a set of implied semantics. Thus, taking the concept of Pthreads, we have used the 'C' technology here while developing the algorithms & the codes. Multiple threads are spawned and fed with the search query. These run independently and perform the search. The results are finally joined together [21].

5. CLUSTERING PROCESS

Clustering is the process of organizing objects into groups whose members are similar in some way. A cluster is a collection of objects which are "*similar*" to each other and are "*dissimilar*" to the objects belonging to other clusters. A rough but widely agreed upon framework is to classify clustering techniques as Hierarchical clustering and partitional clustering based on the properties of the generated clusters.

4 different methodologies are used in this technique, viz., searching, clustering, inverted index & the pre-processing (stopping & stemming) as shown in the Fig. 1. Clustering algorithms arranges data items into several groups so that similar items fall into the same group. This is done without any suggestion from an external supervisor, classes and training examples are not given a priori. Most of the early cluster analysis algorithms come from the area of statistics and have been originally designed for relatively small data sets [18].

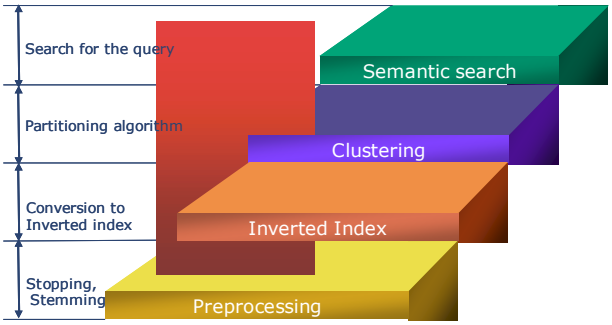

Fig. 1: The different phases of clustering process

In the recent years, clustering algorithms have been extended to efficiently work for knowledge discovery in large databases and some of them are able to deal with high-dimensional feature items. When used to classify large data sets, clustering algorithms are very computing demanding and require high performance machines to get results in reasonable time.

Experiences of clustering algorithms taking from one week to about 20 days of computation time on sequential machines are not rare. Thus, scalable parallel computers can provide the appropriate setting where to execute clustering algorithms for extracting knowledge from large scale data repositories [18].

The different approaches of clustering used in the search process is best explained in the form of a diagram shown in the Fig. 2.

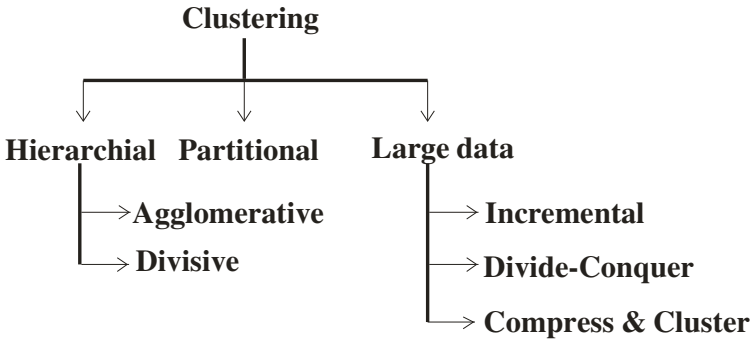

Fig. 2 : Different approaches of clustering process

6. DEVELOPMENT OF THE LEADER'S ALGORITHM

Leader's algorithm [4] is a very simple incremental developmental clustering algorithm. It requires only a single data base scan. It can be used for clustering of numerical data sets and also sequential data sets.

The time complexity of the leaders algorithm is O(nd),
where

d = Dimensionality of the pattern.
n = Total number of patterns.

The space complexity of the algorithm is = O(Ld),
where

L = Number of Leaders.
d = Dimensionality of the pattern.

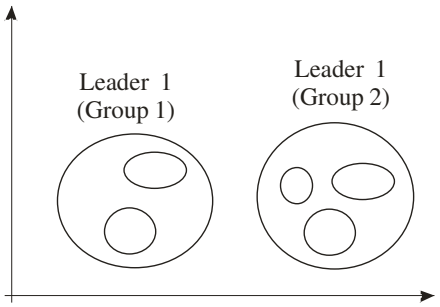

Fig. 3 : Diagram demonstrating the leader's algorithm

Consider some points in space. We choose the first point that we encountered as a leader point. The leader point is compared with all the other points. If the points lie within a specific threshold value, then the groups of points are clustered to form a group. Every group contains a group leader. Similarly, viz., group 1 & the group 2. The different clusters are formed as shown in the Fig. 3.

Leader's algorithm is as follows:

1. Fix an optimal threshold value.
2. Scan the first point and name it as the 'leader'.
3. Scan the next point and compute its distance from the leader point using the Hamming Distance. If the distance is found to be within the threshold value then group these two points and form a cluster.
4. If the point falls out of region of the threshold value, fix that point as next leader. Scan the new point & compute its distance with the leaders and accordingly group the point into respective clusters until next leader.
5. Repeat the entire process until next leader. Leader's algorithm is chosen because it requires only a single scan of the database and so scales up well to deal with large scale problems.

7. SEARCHING PROCESS

Searching is done in 3 different kinds of approaches such as

1. Sequential search
2. Search using threads
3. Search using threads on clustered index

The above mentioned processes are further explained in brief as follows.

1. Sequential search is the basic linear search in which the query is searched for in the entire vocabulary.

2. Search using threads is the mechanism where the vocabulary is divided into two halves. Each half is assigned with a thread to search for the query. Parallelism is achieved here & faster response is delivered to the query.
3. The mechanism used in search using threads on clusters is, after obtaining the clusters we assign each cluster with a thread and search for the query. The time taken in all the search techniques are noted and compared in order to obtain time efficiency. Time taken for searching the query is the least in the case of threads on clusters and highest in sequential search.

8. SIMULATION RESULTS

Simulation is carried out using the developed coding in C language. For the data sets, in order to carry out the testing process, the data set taken is the Table of Contents (TOC) of various books taken from the website http://www.books.com belonging to different fields such as Arts, Music, Architecture, Medicine, Engineering, Law, Defense, etc. This TOC is given as an input to the pre-processing module in the developed C algorithm. After running the program, test results are obtained. The evaluation scheme for the test results is done using three methods, viz., sequential search, threaded based search & the cluster based search for the same datas.

The quantitative results of the comparison of sequential search, thread based search and cluster based search with fixed query length is shown in the Table 1. Also, the graph showing data size vs time taken using the 3 types of search is shown in the Fig. 4. It can be observed from the simulation results shown in this table & from the figure that the searching of data sets using clusters takes less amount of time when compared to sequential or the threaded search and thus improves the results further as the related terms are found together. Also, the clustering with parallelism search becomes more cost effective, time effective and the quality of the search is very accurate as we had used the leader's algorithm. The simulation results also show that the proposed concept of clustering and parallelism search becomes more cost effective, time effective and the quality of the search is accurate. The advantage of the used leader algorithm for the determination of the search indices are, viz., one data base scan, efficient & the access time is very fast.

The quantitative results of comparison of sequential search, thread based search and cluster based searches with fixed data size using the proposed algorithm is shown in Table 2 along with the graph showing query size vs. time taken in Fig. 5. It can be observed from the simulation results presented in this table & from the figure that the cluster based search approach has yielded excellent index results (9 to 14) for various word size, thus reducing the search time in the huge amount of data world. Considering the query size, the cluster based search approach has also yielded excellent index results (11 to 13) for various query sizes ranging from 50 to 80.

Data size	Sequential Search	Thread based search	Cluster based search
10docs (VSize = 865 words)	16.4289	18.4221	14.6044
12docs (VSize = 958 words)	27.4127	18.30435	10.4394
15docs (VSize = 1386 words)	15.2173	13.70035	9.2173

Table 1 : Comparison of sequential search, thread based search and cluster based search with fixed query length.

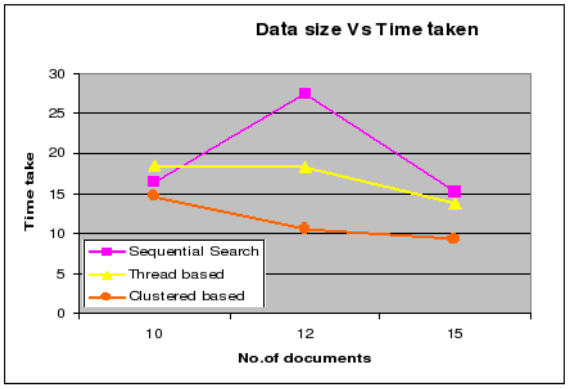

Fig. 4 : Graph showing data size vs time taken

Query size	Sequential search	Thread based search	Cluster based search
50	14.8362	15.5996	13.7597
65	16.6247	15.2558	11.9577
80	16.1377	15.4424	11.8486

Table 2 : Quantitative results of comparison of sequential search, thread based search and cluster based searches with fixed data size using the proposed algorithm.

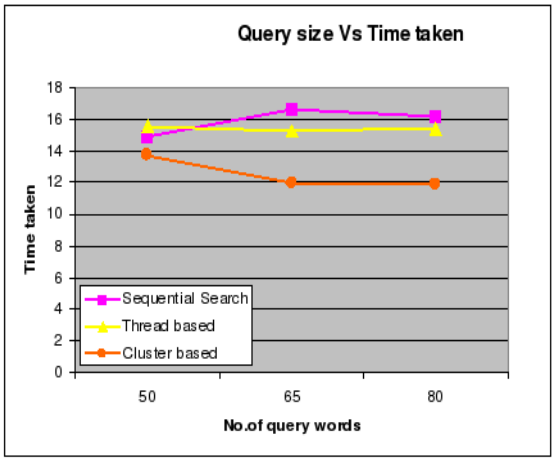

Fig. 5 : Graph showing query size vs. time taken

9. CONCLUSIONS

A novel method of obtaining the clustered distributed indices for efficient text retrieval using threads was presented in this research paper. Simulations in C language were performed to obtain the same. From the statistics and the graphs obtained, it is quite conclusive that search using threads on clusters are faster compared to sequential search. Further, it is justified from the results that searching using clusters takes less amount of time when compared to sequential or the threaded search. Of course, threads helps in improving the performance of the text retrieval as it takes less amount of time to search for the query. But, the cluster based search using threads further improves the results further as related terms are found together.

Our results also show that the proposed concept of clustering and parallelism search becomes more cost effective, time effective and the quality of the search is accurate. The advantage of the used leader algorithm for the determination of the search indices are, viz., one data base scan, efficient & the access time is very fast. For the data size, the cluster based search approach has yielded excellent index results (9 to 14) for various word size, thus reducing the search time. Considering the query size, the cluster based search approach has also yielded excellent index results (11 to 13) for various query sizes ranging from 50 to 80. Sequential search takes more amount of time in the worst case (if the word is not present in the vocabulary). For smaller data sets, sequential search may work faster. In practical scenarios, very large data sets are being used in the world wide web (www). Hence, we could infer that clustered based thread search takes the least amount of time and is considered to be most efficient, which is the highlight of the research work undertaken in this paper.

SCOPE FOR FUTURE WORK

We have used leader's algorithm for the clustering process. Many different clustering algorithms such as *K*-Means, BIRCH, etc., could be used for clustering and then searching on these clustering algorithms can be done. Semantic search is also a good direction to explore.

ACKNOWLEDGMENTS

We would like to acknowledge Shri. A.S. Chinnaswamy Raju, Managing Trustee, Atria Institute of Tech. for providing the facilities for carrying out the research work in this renowned

institution, the AIT. We also thank all our colleagues of AIT & CIT directly or indirectly for helping us while developing this paper. Not but least, we also thank Prof. M.N.Narasimhamurthy of Indian Institute of Science, Bangalore for helping us during the various stages of the work.

REFERENCES

[1]. Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze, "An introduction to information retrieval", *Cambridge University Press*, England, UK, Aug. 30, 2007.

[2]. Alistair Moffat, Justin Zobel, "Self-indexing inverted files for fast text retrieval", Feb. 1994.

[3]. Narasimha Murty M., Anil K Jain, "Knowledge-based clustering scheme for collection management and retrieval of library books", *Jr. of Pattern Recognition*, 3 Jan. 1995.

[4]. Vijaya P.A., M. Narasimha Murthy, D.K. Subramanian, "Leaders-subleaders : an efficient hierarchical clustering algorithm for large data sets", *Pattern Recognition Letters*, Mar. 2004.

[5]. William B. Frakes, Ricardo Baezar-Yates, "Information Retrieval Data Structures & Algorithms", *Prentice Hall PTR*, New Jersey, USA.

[6]. Jiawei Han, Micheline Kamber, "Data mining Concepts and Techniques", *2nd Edition*, *Text Book*.

[7]. http://portal.acm.org/citation.cfm?id=979920

[8]. http://tartarus.org/~martin/PorterStemmer/

[9]. http://en.wikipedia.org/wiki/index

[10]. http://portal.acm.org/citation.cfm?id=979920

[11]. http://www.resample.com/xlminer/help/HClst/HClst_intro. html

[12]. http://ecommerce.hostip.info/pages/924/Search-Engine-Strategy.html

[13]. Hisashi Koga, Tetsuo Ishibashi and Toshinori Watanabe, "Fast Hierarchical Clustering Algorithm using Locality Sensitive Hashing, Graduate Schools of Information Science", *University of Electro-Communications*.

[14]. Narasimha Murty M., Anil K Jain, "Knowledge-based clustering scheme for collection management and retrieval of library books", *Pattern recognition and Image Processing Laboratory,* Dept. of Computer Science, Michigan State University, East Lansing, MI 48824-1027, U.S.A., 3 Jan. 1995.

[15]. Anna Formica, "Concept Similarity by evaluating information content ad feature vectors: a combined approach", *Communications of the ACM*, Vol. 52, No. 3, Mar. 2009.

[16]. Foti D., D. Lipari, C. Pizzuti and D. Talia, "Scalable Parallel Clustering for Data Mining on Multi - computers", *Jr. paper.*

[17]. Judd D., P. McKinley, A. Jain, "Performance Evaluation on Large-Scale Parallel Clustering in NOW Environments", *Proceedings of the Eight SIAM Conf. on Parallel Processing for Scientific Computing*, Minneapolis, Mar. 1997.

[18]. Stoffel K and A. Belkoniene. Parallel K-Means Clustering for Large Data Sets", *Proceedings Euro-Pa'99*, LNCS 1685, pp. 1451-1454, 1999.

[19]. Justin Zobel and Alistair Moffatacm, "Inverted Files for Text Search Engines", Computing Surveys, Vol. 38, No. 2, Article 6, Jul. 2006.

[20]. Zobel, J., Moffat, A. & Sacks-Davis, R., "An efficient indexing technique for full-text database systems", *Proc. VLDB Int. Conf. on Very Large Databases*, L.-Y. Yuan, Ed. Morgan Kaufmann, Vancouver, pp. 352–362, 1992.

[21]. Mahesh V Joshi, Eui Hong Sam Han, George Karypis and Vipin Kumar, "Parallel Algorithms in Data Mining", Conf. paper.

[22]. Manber, U. and Myers, G., "Suffix arrays: A new method for on-line string searches", Proc. of the 1st ACM-SIAM Symposium on Discrete Algorithms, ACM Press, New York, NY, 319–327, 1990.

[23]. Salton, G., "Information Retrieval: Data Structures and Algorithms", Addison-Wesley, Reading, Massachussetts, 1989.

[24]. Faloutsos, C. and Christodoulakis, S., "Signature files: An access method for documents and its analytical performance evaluation", ACM Trans. on Office Information Systems 2, 4 (October), ACM Press, New York, NY, 267–288, 1984.

Mr. M. Basavaraju completed his Masters in Engineering in Electronics and Communication from University Visvesvaraya College of Engg. (Bangalore), Bangalore University in 1990, & B.E. from Siddaganga Institute of Technology (Tumkur), Bangalore University in the year 1982. He has got a vast teaching experience of 22 years & an industrial experience of 6 years. Currently, he is working as Professor and Head of Computer science & Engg. Dept., Atria Institute of Technology, Bangalore, India. He is also a research scholar in Coimbatore Inst. of. Tech., Coimbatore, doing his research work & progressing towards his Ph.D. in the computer science field from Anna University Coimbatore, India, His research interests are Data Mining, Computer Networks, Parallel computing.

Dr. R. Prabhakar obtained his B. Tech. degree from IIT Madras in 1969, M.S. from Oklahoma State University, USA and Ph.D. from Purdue University, USA. Currently he is professor of Computer of Science and Engineering and Secretary of Coimbatore Institute of Technology, Coimbatore, India. His areas of specialization include Control Systems, CNC Control, Robotics, Computer Graphics, Data Structures, Compilers, Optimization. He has published a number of papers in various national & international journals, conferences of repute & is guiding a couple of students who are progressing towards the Masters & Doctoral degrees.