A New Context Based Indexing in Search Engines Using Binary Search Tree

Aparna Humad
Department of Computer science and Engineering
Mangalayatan University, Aligarh, (U.P)

Vikas Solanki
Department of Computer Science and Engineering
Mangalaytan University, Aligarh,(U.P)

Abstract- Indexing in search engines in is an active area in current researches. The main aim of search engines is to provide most relevant documents for a user query in minimum possible time. So fast accessing the keywords stored in an index is a major issue for performances of Web Search Engines. Indexing is performed on the web pages after they have been stored into a repository by the crawler. The existing architecture of search engine shows that the index is built on the basis of the terms of the document and consists of an array of the posting lists where each posting list is associated with a term and contains the term as well as the identifiers of the documents containing that term. The current search engines use terms to retrieve documents from it. This paper proposes an indexing structure in which index is built on the basis of context of the document rather than on the basis of terms. The context of the documents in the repository is being extracted by the indexer using thesaurus. Then documents are indexed according to their respective context. The proposed system uses a BST Binary search tree to store keywords in the index. BST will provide a faster access to the index and context will improve the relevancy in the search results returned to the user.

Keywords – context , indexing, information retrieval, search engine, thesaurus.

I. INTRODUCTION

The internet contains hundreds of thousands of electronic collections that often contain high quality information. The basic aim is to select the best collection of information for a particular information need. The indexing phase of search engines can be viewed as a Web Content Mining process. Starting from a collection of unstructured documents, the indexer extracts a large amount of information like the list of documents, which contain a given term. It also keeps account of number of all the occurrences of each term within every document. This information is maintained in an index, which is usually represented using an inverted file (IF). IF is the most widely adopted format for this index due to its relatively small size occupancy and the efficiency involved in resolution of the keywords based queries. The index consists of an array of the posting lists where each posting list is associated with a term and contains the term as well as the identifiers of the documents containing the term. The term based index seems to be less efficient due to two information retrieval problems: polysemy (means a word has multiple meanings) and synonymy (means that multiple words having the same meaning). Thus the significance of term for building the index is reduced and the emphasis is laid on the context of the document. Context provides extra information to improve search result relevance.
Some index organization related techniques which has been proposed are also discussed here:

In [12], the authors introduce a double indexing mechanism for search engines based on campus Net. The Campus Net Search Engine (CNSE) is based on full-text search engine, but it is not general full-text search engine as it is basically a private net. The CNSE consists of crawl machine, Chinese automatic segmentation, index and search machine. They proposed double indexing mechanism, which means, it has document index as well as word index. The so-called document index is based on the documents do the clustering, and ordered by the position in each document. In the retrieval, the search engine first gets the document id of the word in the word index, and then goes to the position of corresponding word in the document index. Because in the document index, the word in the same document is adjacent, the search engine directly compares the largest word matching assembly with the sentence that users submit. The mechanism proposed by them seems to be time consuming as the index exists at two levels.

Another work proposed was the reordering algorithm [2] which partitions the set of documents into k ordered clusters on the basis of similarity measure. According to this algorithm, the biggest document is selected as centroid of the first cluster and n/k1 most similar documents are assigned to this cluster. Then the biggest document is selected and the same process repeats. The process keeps on repeating until all the k clusters are formed and each cluster gets completed with n/k documents. This algorithm is not effective in clustering the most similar documents. The biggest document may not have similarity with any of the documents but still it is taken as the representative of the cluster.

In [6] authors propose threshold based clustering algorithm in which the number of clusters is unknown. However, two documents are classified to the same cluster if the similarity between them is below a specified threshold. This threshold is defined by the user before the algorithm starts. It is easy to see that if the threshold is small, all the elements will get assigned to different clusters. If the threshold is large, the elements may get assigned to just one cluster. Thus the algorithm is sensitive to specification of threshold. The given paper discusses the context based indexing structure in which the index is being created on the basis of the context of the terms in the documents rather than on the basis of terms itself.

II. Proposed Algorithm

In this paper a new scheme for indexing has been proposed. The architecture of the proposed system is shown in Fig-1.The main components of the proposed system are as follows:

A. Crawler

This component will download the web pages from the world wide web and stores in a repository for further processing by indexer.

B. Indexer

This module will extract the web pages from the repository and parse these documents to extract a list of keywords. It also find the context of the document from Context Finder module. It stores the keywords in the binary search tree. It also stores the Doc-id (Document id) and it's context along with keyword in the BST.

C. Context Finder

This module find the context of a document. It takes the help of thesaurus to find the context of a document.

D. Thesauras

It is a dictionary of words available on the world wide web from thesaurus.com which contains the words as well as their multiple meanings. This information will be helpful for the context finder module to find the context of the document.

E. BST Binary Search Tree

It is data structure that will store the complete index. Fast access is the main characteristic of a BST. So BST will improve the efficiency of the indexing system to provide faster access to the ranking system to extract the list of documents suitable for a user query. The nodes of the BST will store the list of available context for a keyword. When a keyword is to be stored then indexer first search keyword in BST. If a node already exist for the keyword then indexer will add a new entry in the document list of the context. The document list for a context will store the doc-id and frequency of the keyword in that document.

F. Query Interface

This module read the query from the user. It access the indexer to find the documents which are relevant for the user query. It also provide an interface to show the searched results to the user.

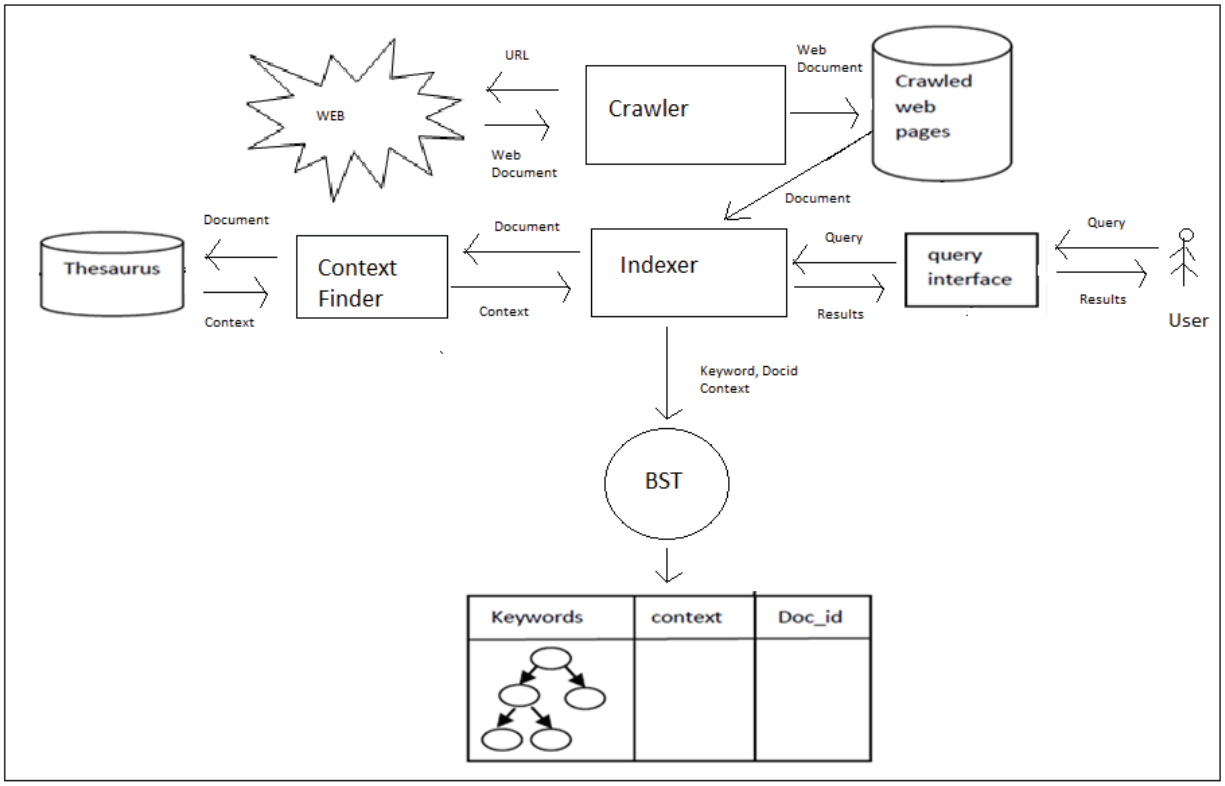

Figure-1 Architecture of Context Based Indexing in Search Engines Using Binary Search Tree

The algorithm used to create the BST is as follows:

Step1: Pre-process the crawled web documents and extract the keyword along with their frequency of occurrence.
Step2: For every keyword found in document search the keyword in the Binary Search Tree, until a similar word is found.
Step3: If keyword already exists in BST then search the context in the list of available context for BST node. Make an entry of doc-id and its frequency in the context list of the document. If context of the document is not found then create a new entry of context and add the doc-id and it's frequency in this new context.
Step4: If search is not a successful, create a node containing the following fields (left-child, keyword, right-child, link).The link is pointer variable which points to the database where the context of keyword and the corresponding document_id is stored. Arrange the node in the Binary Search Tree.
Step5: Repeat step 2, 3,4 and 5 till all the extracted keywords of the document has been arranged.

This algorithm will arrange the keywords according to their context. It will create a Binary Search Tree(BST) . The nodes of the BST store the keyword and a list of context in which that keyword is used. Now when the user fires the query with context explicitly specified, then the index is being searched, reducing its search time to half of the linear search. Thus this new Binary Search Tree indexing technique provides a fast access to document context and to find the list of documents related to user query. The whole system will improve the precision and recall performance measures of the search engine.

III. Experiment and Result

The proposed work has been implemented in java using NetBeans open source IDE. The Fig-2 shows

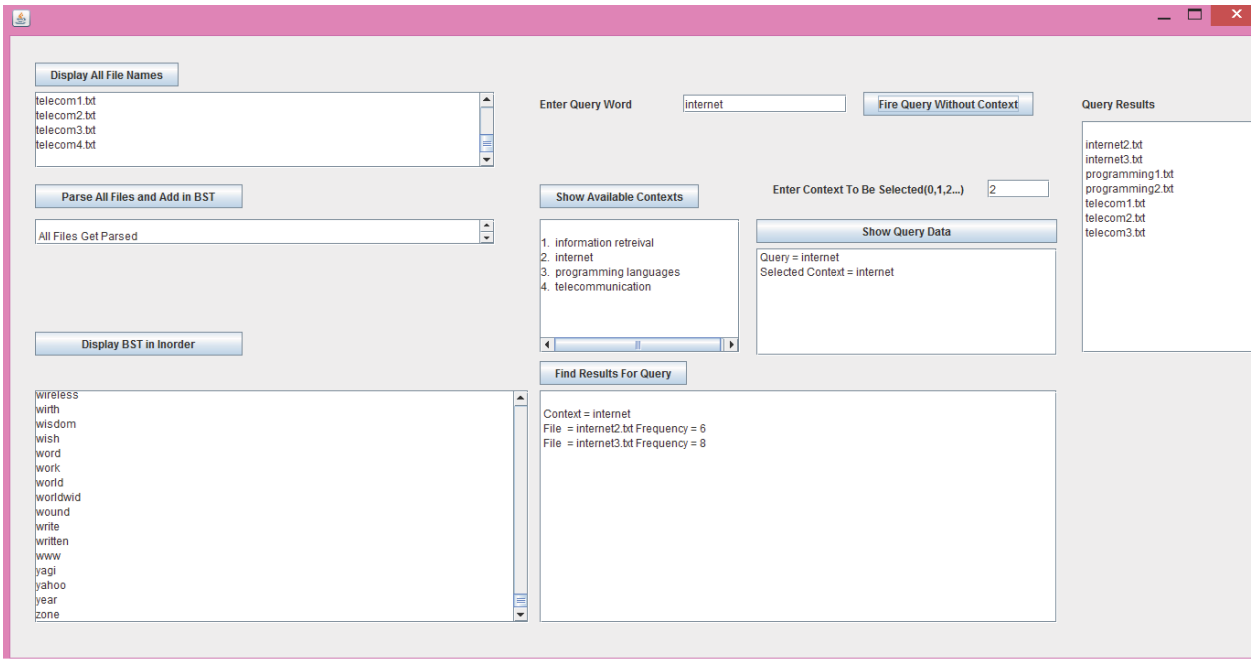

Figure-2 : Snap shot of experiment performed.

The proposed method has been implemented in java and a snapshot is shown in Fig-2. It has been implemented for a sample corpus of 11 text files. The files belong to different contexts. In the experiments 2 files belongs to context "information retrieval", 3 files belongs to context "internet", 2 files belongs to context "programming languages" and 4 files belong to context "Telecommunication". These files has been parsed to retrieve tokens. Stop words has been removed and stemming has been performed on each token. Then the tokens has been inserted in Binary Search Tree BST. Then a query has been fired. The user has to specify the query and the context of the query. A list of available contexts has been provided to the user.

For example :
Query : internet
Context : internet
Result : Two files has been retrieved which belongs to context "internet".

When the same query has been fired on the whole index without entertaining the context, seven files has been retrieved. Two document from the "internet" context, two from "Programming Languages" context and three from "Telecommunication" context..

Comparative Analysis
The results has been compared for precision value for both context based searching and normal searching.
Precision is the fraction of the documents retrieved that are relevant to the user's information need.

Precision = (|{Relevant Documents}∩{Retrieved Documents}|) / |{Retrieved Documents}|
Without context:
Relevant Documents = 2
Retrieved Documents = 7
(|{Relevant Documents}∩{Retrieved Documents}| = 2
Precision = 2/7 = 0.28

With Context
Relevant Documents = 2
Retrieved Documents = 2
Precision = 2/2 = 1.0
So almost all the documents retrieved will be relevant to user in context based retrieval.

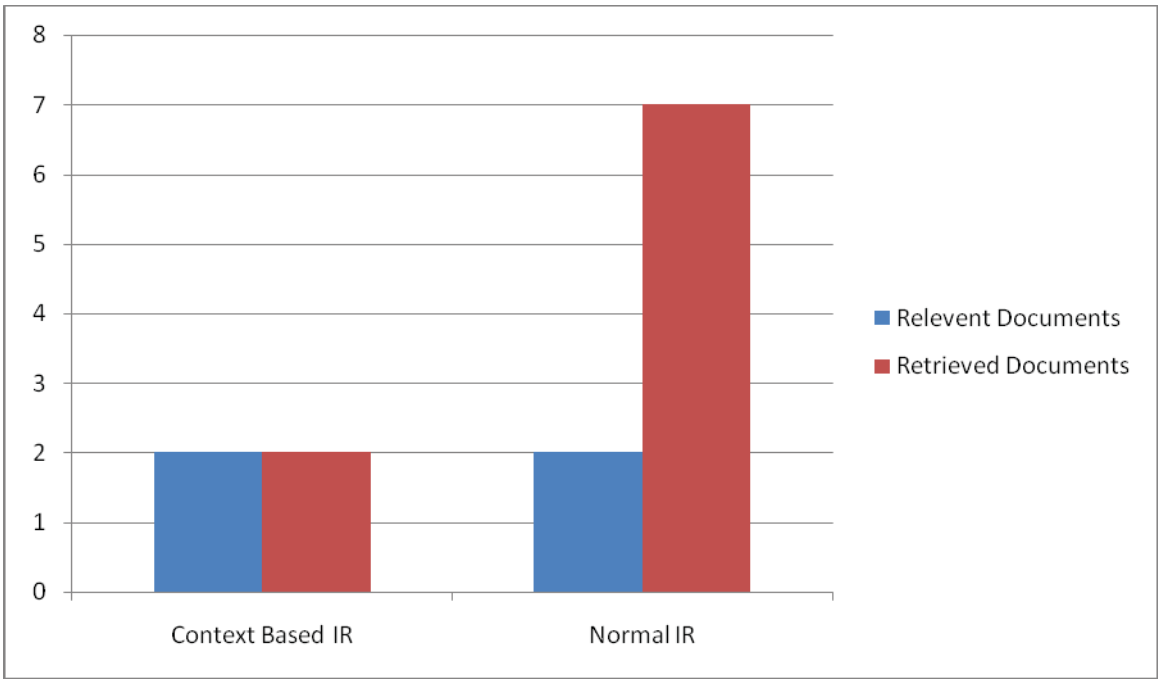

Figure-3: Comparative Analysis of Context Based IR Indexed by BST and Normal IR.

IV.CONCLUSION

In this paper a new architecture for Context Based Indexing in Search Engines Using Binary Search Tree has been proposed. It uses the advantage of fast searching of BST. It also stores the context information for every keyword and availability of the keyword in the documents. This new architecture will provide a list of relevant document (With the help of context) for the user query in minimum time (with the help of BST). Thus the new system will improve the precision value of the search engine and also provide the search results in minimum time. However this architecture have to be implemented on a corpus with large number of documents in different contexts.

REFERENCESS

[1] Steve Lawrence, "Context in Web Search", IEEE Data Engineering Bulletin, 2000.
[2] Fabrizio Silvestri, Raffaele Perego and Salvatore Orlando. Assigning Document Identifiers to Enhance Compressibility of Web Search Engines Indexes. In the proceedings of SAC, 2004.
[3] S.Brin and L.Pagse.The Anatomy of a Large-Scale Hypertextual Web Search Engine. In the 7th International WWW Conference, (WWW7), pp. 107-117, Brisbane, Australia.
[4] Van Rijsbergen C.J. Information Retrieval. (Butterworth 1979).
[5] Soumen Chakrabarti. Mining the Web Discovering Knowledge from Hypertext Data. (Morgan Kaufmann Publications, 2003).
[6] Oren Zamir and Oren Etzioni. Web Document Clustering: A feasibility demonstration. In the proceedings of SIGIR, 1998.
[7] A. Jain and R. Dubes. Algorithms for Clustering Data.(Prentice Hall, 1988)
[8] Berners-Lee, T., Hendler, J. and Lassila, O., "The Semantic Web," Scientific American.284(5):35-43, 2001.
[9] O. Zamir, O. Etzioni, O. Madanim, and R.M. Karp, "Fast andIntuitive Clustering of Web Documents," Proc. Third Int'l Conf. Knowledge Discovery and Data Mining, pp. 287- 290, Aug. 1997.
[10] Wang Jicheng, Huang Yuan, Wu Gangshan and Zhang Fuyan, 'Web Mining: Knowledge Discovery on the Web' ,IEEE (1999).
[11] Frawley, W., Piatetsky-Shapiro, G., and Matheus, C., Knowledge Discovery in Databases: An Overview. Ai Magazine, Vol. 13 (1992), pp.57-70.
[12] Changshang Zhou, Wei Ding, Na Yang, Double Indexing Mechanism of Search Engine based on Campus Net, Proceedings of the 2006 IEEE Asia-Pacific Conference on Services Computing (APSCC'06)
[13] Quan, T. T., Hui, S. C., Fong, A. C. M., and Cao, T. H. (2004). Automatic generation of ontology for scholarly semantic Web. In: Lecture Notes in Computer Science. Vol. 3298. (pp. 726–740).