CONTEXT INDEXING IN SEARCH ENGINE USING BINARY SEARCH TREE

Charu Kathuria
Information Technology
Lingaya's University
Faridabad, India
Charu.kathuria05@gmail.com

Goutam Datta
Information Technology
Lingaya's University
Faridabad, India

Vanditaa Kaul
Computer Science & Engineering
B.S Anangpuria Institute of Technology and Management
Faridabad, India

***Abstract*-Indexing in search engines is an active area of current researches and the aim of search engines is to provide the most relevant documents to the users in least possible time. In this paper a context based indexing is proposed where the index list stores the available context along with the keywords. Thus the information will be more relevant as the context of the topic is also made available to the retrieval system. For indexing the keyword extracted from the web documents along with their contexts are stored in Binary Search Tree (BST), to enhance the performance of the search engine.**

***Index Terms*-Binary Search Tree (BST), Search engine, World Wide Web, Context Indexing, Indexing Structure.**

I. INTRODUCTION

WWW on the Web is a service that resides on computers that are connected to the Internet and allows end users to access data that is stored on the computers using standard interface software.

Search engine is a computer program that searches for particular keywords and returns a list of documents in which they were found, especially a commercial service that scans documents on the Internet. A traditional search engine comprises of three core components: download, index and search. Out of three the "download" is the most important activity wherein the web is traversed to download web pages by following links. A copy of all visited pages is stored for later use.

Today the size of the web is increasing with a tremendous rate. So it is very difficult for web user to fulfill his or her information need. Since the user enters only a combination of keywords to search information without giving a thought about the context, search engines therefore search without concerning the user's context of search; providing results which may or may not fulfill the requirements.

The basic aim is to select the best collection of information according to users need. The existing search engines adopt different strategies for computing the words frequency in the web documents. If higher frequency words match with the topic keyword, then the document is considered to be relevant. But they generally do not analyze the context of the keyword in the web page before they download it.

Binary search tree is most basic, nonlinear data structure in computer science that can be defined as "a finite set of nodes that is either empty or consists of a root and two disjoint subsets called left and right sub-trees". Binary trees are most widely used to implement binary search algorithm for the faster data access.

II. RELATED WORK

In this section, a review of previous work on index organization is given. In this field of index organization and maintenance, many algorithms and techniques have already been proposed.

C. Zhou, W. Ding and Na Yang [1], the paper introduces a double indexing mechanism for search engines based on campus Net. The proposed mechanism has document index as well as word index. The document index is based on, where the documents do the clustering, and ordered by the position in each document. During 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, seems to be time consuming as the index exists at two levels.

Another literature CDFC [2] [3] has identified the need of contextual information to be sent along with the query terms and proposed method to represent the context in the form of augmented hypertext documents. It is an agent-based context driven focused crawler. It gets the user input and then presents the user different meaning of the input keyword for further selection of specific context. This is done by storing the volatile information and the context and the keywords in separate file say TVI (table of volatile information) and TOC (table of context) for every augmented hypertext document. These files are retrieved by agents in spite of the whole documents and are indexed by crawler. TOC information is presented to the user for their specific search.

Another work proposed was the reordering algorithm [4] 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.

Another proposed work on context based indexing in search engines using ontology by P. Gupta and A. K. Sharma [5]. The index construction is done on the basis of the context using ontology. The context repository, thesaurus and ontology repository are used by the indexer to identify the context of the document.

The overlook at the available literature reveals that there is a need of a technique to organize the keywords and their contexts in a better fashion as arranging in a linear fashion makes searching of a document time consuming.

III. PROPOSED WORK

The architecture of the proposed context based indexing in information retrieval system is as shown in the following Figure (1).

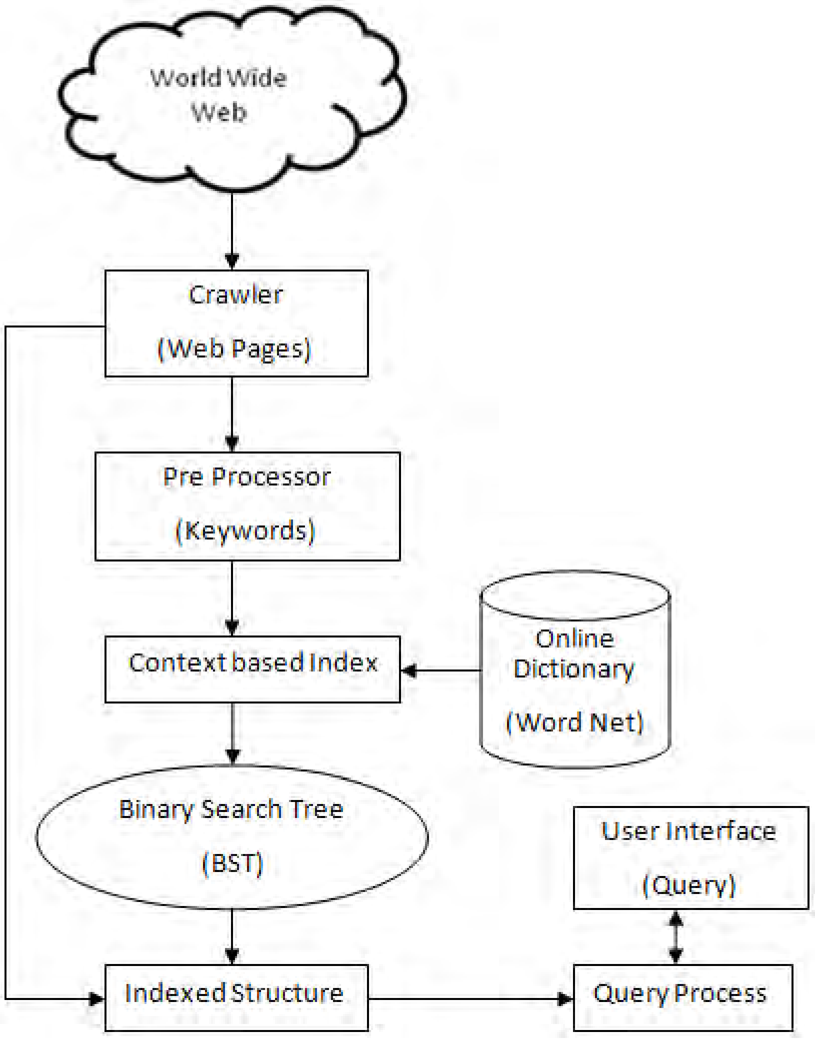


Figure (1): Architecture of Context Based Indexing

In this architecture of context based indexing, web pages are stored in the crawled web page repository. The indexes hold the valuable compressed information for each web page. The preprocessing steps are performed on the documents (i.e. stemming as well as removal of stop words). The keywords are extracted from the document, and their corresponding multiple contexts are identified from the Word Net. Indexer maintains the index of the keyword using the Binary Search tree. The user submits the query through the search engine interface and the relevant information is made available to the user after searching the results in the index.

Indexed structure for binary search tree

The design of data structure used to store the documents is shown in the figure (2).

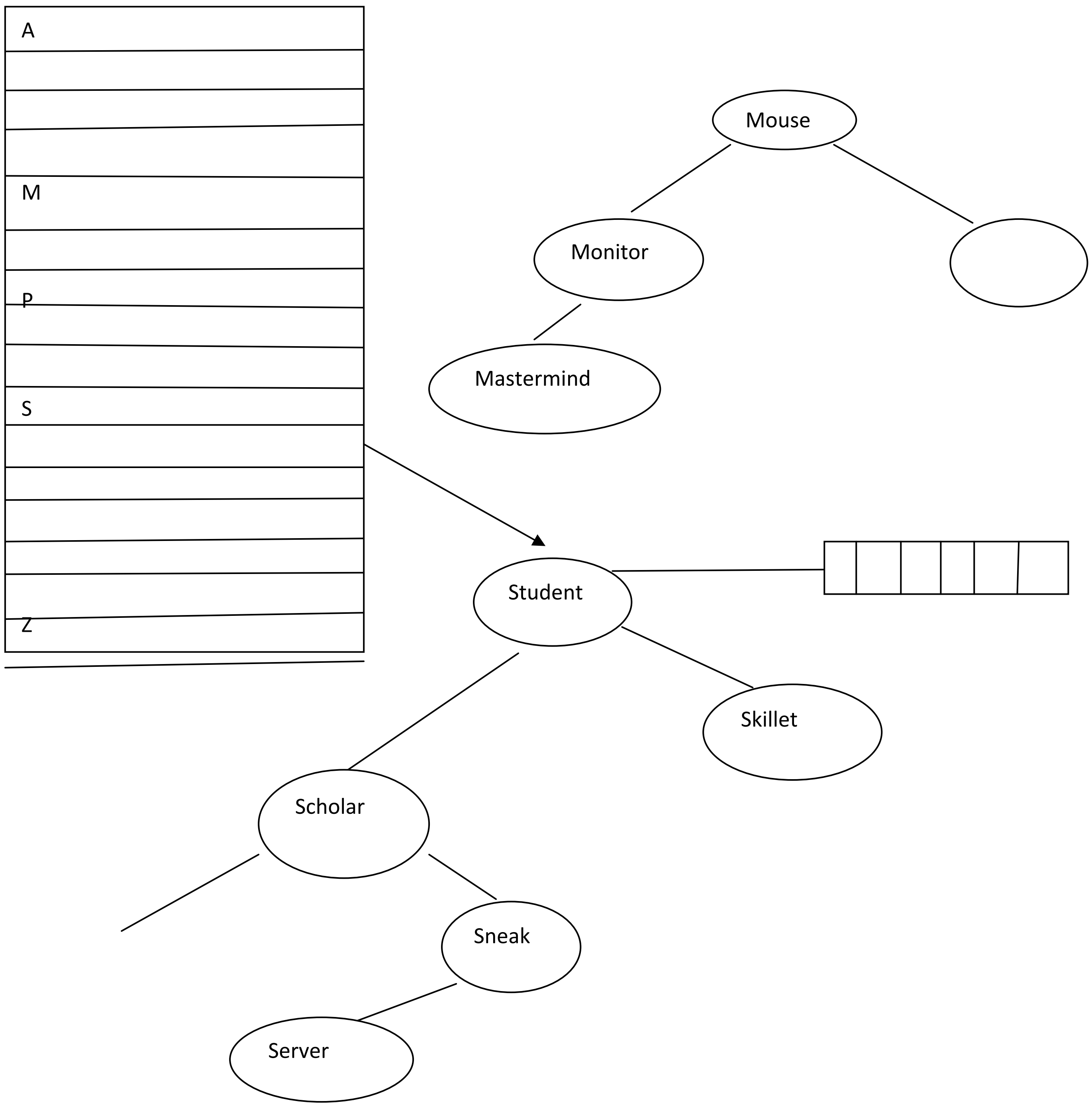


Figure (2) Data Structure used to store document keywords

The data structure used to store the documents is as shown in the figure (1). Documents are arranged by the keywords it contains and the index is maintained in lexical order. For every alphabet in the index there is one BST (Binary search tree) containing the keywords with the first letter matches with the alphabet. Each node in the BST points to a structure that contains the list of contextual meanings corresponding to that keyword and contains the pointers to the documents that matches the particular meaning.

The structure is designed keeping in mind the quick response to the user query. As the query comes in form of keyword, first the index is searched by the beginning letter of the keyword. After that corresponding BST is searched in search insert fashion for a match of the keyword, if found the list of contextual meanings is displayed to the user. A selection from that is received from the user and then the search progress only for the matched documents related to the user specified meaning of the keyword. Each meaning again is node with two fields, meaning and the list of pointers. These pointers points to the documents stored in the local database with unique Id's and these Id's are treated as pointers in the node of structure linked with every keyword node of BST.

The data/information in the structure is updated as soon as the Word Net is updated with a new context for a keyword and the new document found that contain the existing/new keywords. Though there is some redundancy of the information stored in this structure but it serves the query faster. Documents are stored in a separate structure may be in ring files format.

Detailed structure of a node

This section provides details of the context based indexing structure to serve as a reference for implementation purposes. The indexing information is stored in the tree nodes themselves in the form of meaning/context tables.

Figure (3) illustrates the information stored in one of the nodes given in example.

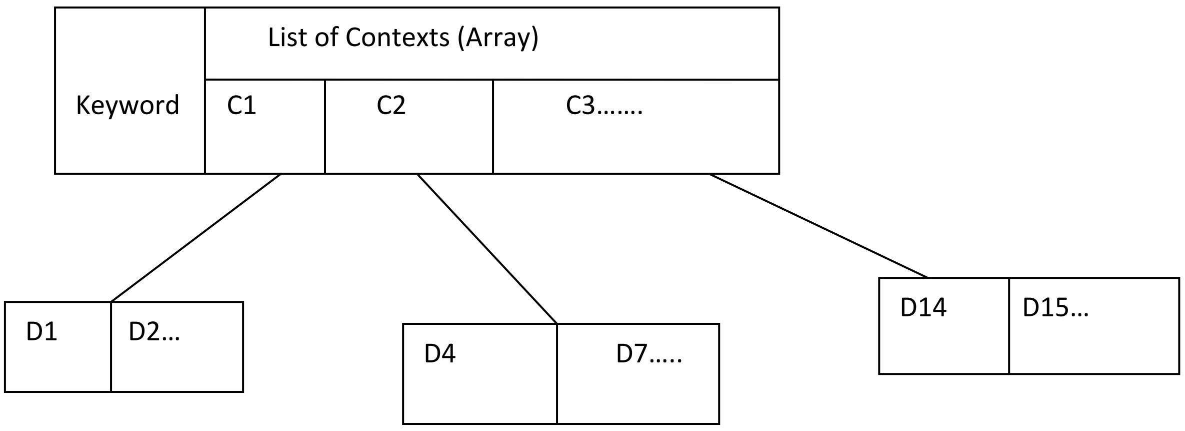


Figure (3): Structure of a node

Here,

Keyword – is the keyword that appear in some or more documents in local database and that will match the user query keyword.

List of Contexts – is the list of all different usage/senses of the keyword obtained from the WordNet.

C1 – stands for the contextual sense 1

With each Contextual sense (C) a list of pointers to the documents in which this C appears is associated. Where,

D1 – stands for the pointer to document 1

Basically the structure maintained in each node is a table of contexts/meanings with each meaning having list of pointers to the corresponding documents.

Since the structure is a tree each node maintains a list of contextual meaning for that node. This list helps in representing the user different meanings/senses of the keyword represented by the node.

The tree is arranged in BST fashion. At the time of creation the first keyword that comes becomes the root of the tree and other keywords started with that particular alphabet are arranged according to the value of that keyword.

In the structure shown in figure (2) it have 26 BST trees each corresponding to one English alphabet.

Each node in the alphabet tree represents the search keyword started from that alphabet. It can be noted that each node has an associated list of meanings in the table form where each meaning has a list of pointers that are pointing to the document specific to that particular meaning. This pointer to document is basically the document ID that is unique for each document. This document ID is generated for each document at the time of storage of the document in another structure for reference.

It is noted that same document can match to satisfy a number of query keywords so the document should be unique and all the documents should be stored separately in a different structure.

Algorithm for indexing

Klist [] – This array contains the keywords from the document.

```
Algo-Create-BST
{
```

```
For (each document in local database)
{
 Klist [] = all the keywords from a document.
For (i=0;i<=length of Klist[];i++)
    {
      If (Klist[] ==’A’ to ‘Z’)
            a)  The corresponding BST is selected
            b)  If (no BST with this alphabet exists)
                Create a new BST with this alphabet
                Root [BST] = Keyword
                Append the node with corresponding different contextual meaning and a pointer to the
                document
          Else
          Compare the Klist [] with root node & further down the tree to do the appropriate insertion in BST with
          corresponding list of senses and pointer to document.
Else
              If Klist [] ==value(node[i]) then
                        Add the pointer to this document in the list of pointers to documents
     }
}
```

Steps to construct the Index Structure

1. For each document in local databases extract all the keyword from document.
2. For each keyword find a match with 26 English alphabets.
3. The corresponding BST(binary search tree) is selected for further processing.
4. If the keyword is the first keyword in the index that start with that alphabet the new BST is created and root of the tree is assigned this keyword and at the same time node is attached containing contextual meaning from Word Net and pointers to the corresponding documents in repository.
5. Else if the BST for the alphabet exist the keyword value is compared with the root node value to do the appropriate insertion in BST with the list of meaning/senses and pointers to the documents.
6. Else the keyword is matched with some entry in BST, the list of pointers to documents is updated with this new document.

Steps to search the index to resolve a query

1. For the query keyword given by the user, search the index to get match with the first alphabet of the keyword
2. The corresponding BST is selected for further searching.
3. If a match is found with some entry, corresponding list of meaning i.e. C1, C2, C3…etc is displayed to the user to get the user selection, after getting a specific choice from the user the corresponding list of pointers is accessed to get the documents from the repository and finally displayed to the user as final result for the query.
4. Else if the keyword is not found in the corresponding BST, the appropriate insertion is done in the BST and no match found is displayed to the user.

IV. RESULTS

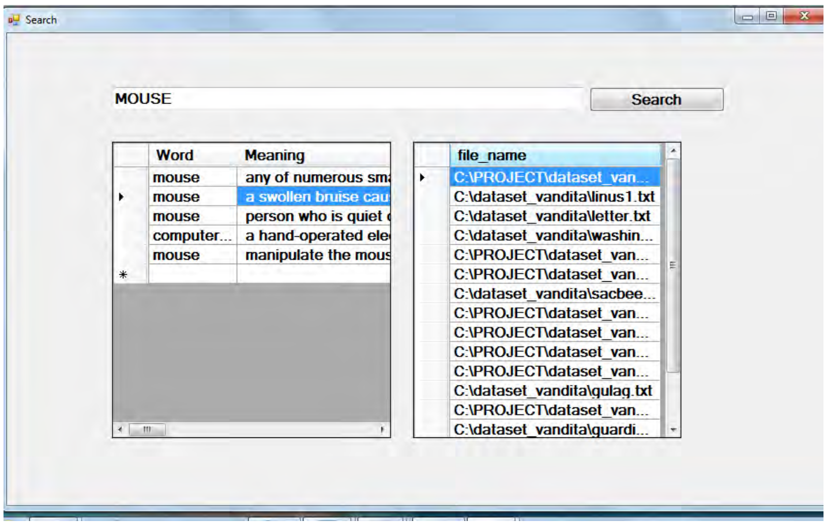
Now, using the above algorithm search engine is implemented as follows.
The interface includes

1. One textbox to enter the keyword
2. Two grid views
    - Display context related to keywords
    - Locations related to the selected context from first grid view.
3. Selected file location displayed

Steps to be followed

- User enters the desired keyword in the textbox. Example: Mouse
- Now click the search button
- Different contexts are then displayed in the first grid view
- On selecting the particular context, related document locations are displayed in the second grid view
- Selected file location is then displayed

SCREENSHOT 1:

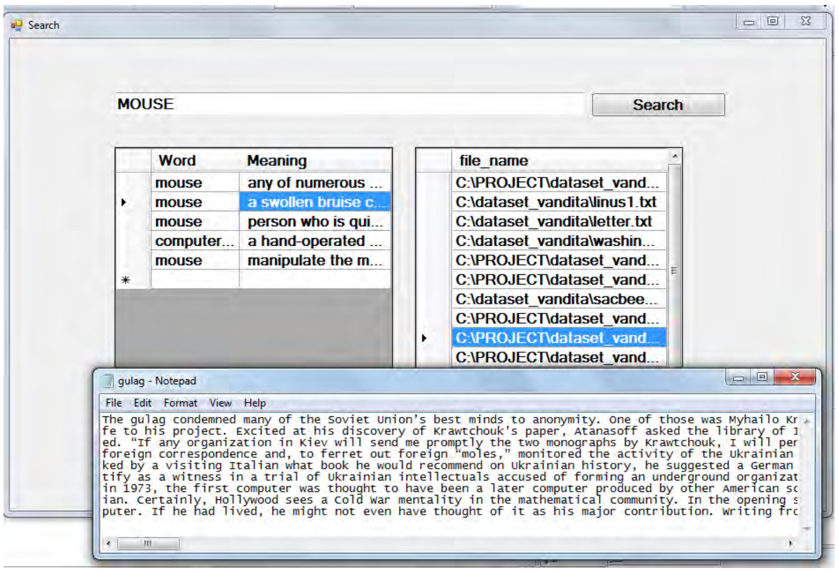


Word	Meaning
mouse	any of numerous sm
mouse	a swollen bruise cau
mouse	person who is quiet
computer...	a hand-operated ele
mouse	manipulate the mous

file_name
C:\PROJECT\dataset_van...
C:\dataset_vandita\linus1.txt
C:\dataset_vandita\letter.txt
C:\dataset_vandita\washin...
C:\PROJECT\dataset_van...
C:\PROJECT\dataset_van...
C:\dataset_vandita\sacbee...
C:\PROJECT\dataset_van...
C:\PROJECT\dataset_van...
C:\PROJECT\dataset_van...
C:\PROJECT\dataset_van...
C:\dataset_vandita\gulag.txt
C:\PROJECT\dataset_van...
C:\dataset_vandita\guardi...

Figure(4): Different context related to a keyword and their location

SCREENSHOT 2:



Word	Meaning
mouse	any of numerous ...
mouse	a swollen bruise c...
mouse	person who is qui...
computer...	a hand-operated ...
mouse	manipulate the m...

file_name
C:\PROJECT\dataset_vand...
C:\dataset_vandita\linus1.txt
C:\dataset_vandita\letter.txt
C:\dataset_vandita\washin...
C:\PROJECT\dataset_vand...
C:\PROJECT\dataset_vand...
C:\dataset_vandita\sacbee...
C:\PROJECT\dataset_vand...
C:\PROJECT\dataset_vand...
C:\PROJECT\dataset_vand...

```
The gulag condemned many of the Soviet Union's best minds to anonymity. One of those was Myhailo Kr
fe to his project. Excited at his discovery of Krawtchouk's paper, Atanasoff asked the library of I
ed. "If any organization in Kiev will send me promptly the two monographs by Krawtchouk, I will per
foreign correspondence and, to ferret out foreign "moles," monitored the activity of the Ukrainian
ked by a visiting Italian what book he would recommend on Ukrainian history, he suggested a German
tify as a witness in a trial of Ukrainian intellectuals accused of forming an underground organizat
in 1973, the first computer was thought to have been a later computer produced by other American sc
ian. Certainly, Hollywood sees a Cold War mentality in the mathematical community. In the opening s
puter. If he had lived, he might not even have thought of it as his major contribution. Writing fro
```

Figure(5): Reaching the specified page

V. CONCLUSION

This paper proposes a technique for indexing the keywords extracted from the web documents along with their context. The Binary Search tree based indexing technique, is able to support dynamic indexing and improves the performance in terms of accuracy and efficiency for retrieving more, relevant documents as per the user's requirements since the context of the various keywords is also stored along with them. Thus, the indexing technique provides a fast access to document context and structure along with an optimized searching.

VI. REFERENCES

[1] Changshang Zhou, Wei Ding and 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), 2006.

[2] N. Chauhan, A. K. Sharma, "Context Driven Focused Crawling: A New Approach to Domain-Specific Web Retrieval", paper presented at *International Conference on information & Communication Technology (IICT)*, July, 2007.Dehradun.

[3] N. Chauhan, A. K. Sharma, "A Framework to Derive Web Page Context from Hyperlink Structure", *International Journal of Information and Communication Technology*, 2008 Vol. 1, No.3/4pp.329-346.

[4] Fabrizio Silvestri, Raffaele Perego and Salvatore Orlando. Assigning Document Identifiers to Enhance Compressibility of Web Search Engines Indexes. In the proceedings of SAC, 2004.

[5] Parul Gupta and A.K.Sharma," Context based Indexing in Search Engines using Ontology", International Journal of Computer Applications, Volume 1 No. 14, pp 49-52, 2010.

[6] Pooja Gupta, Ashok Sharma, J. P. Gupta and Komal Bhatia, "A Novel Framework for Context Based Distributed Focused Crawler (CBDFC)", *Int. J. Computer and Communication Technology, Vol. 1, No. 1, 2009*
[7] Nidhi Tyagi, Rahul Rishi, R.P.Aggarwal, "Context based Web Indexing for Storage of Relevant Web Pages", International Journal of Computer Applications Vol40/2012
[8] Parul Gupta, Dr. A.K.Sharma, "Context based Search engine in Ontology", International Journal Of Computer Application. Vol.1/ No. 14.

AUTHORS

First Author – Charu kathuria, Student(M.Tech), Lingaya's University , charu.kathuria05@gmail.com
Second Author – Goutam Datta, Asst. Proff., Lingaya's University, gdatta1@yahoo.com
Third Author – Vanditaa Kaul, Lecture, B.S.Anangpuria Insitute of Technology & Management. vanditaa.kaul@gmail.com