Context based Web Indexing for Storage of Relevant Web Pages

Nidhi Tyagi
Asst. Prof.
Shobhit University, Meerut

Rahul Rishi
Prof. & Head
Technical Institute of Textile & Sciences, Bhiwani

R.P. Agarwal
Prof. & Head
Shobhit University, Meerut

ABSTRACT
A focused crawler downloads web pages that are relevant to a user specified topic. The downloaded documents are indexed with a view to optimize speed and performance in finding relevant documents for a search query at the search engine side. However, the information will be more relevant if the context of the topic is also made available to the retrieval system. This paper proposes a technique for indexing the keyword extracted from the web documents along with their contexts wherein it uses a height balanced binary search (AVL) tree, for indexing purpose to enhance the performance of the retrieval system.

General Terms
Algorithm, retrieval, indexer.

Keywords
AVL tree, contextual, repository, balance factor.

1. INTRODUCTION
With the rapid growth of the Internet, the World Wide Web (WWW) has become one of the most important resources for obtaining information and one of the most important media of communication. The basic aim is to select the best collection of information according to users need. The existing focused crawlers [1, 2] 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. The subject of context has received a great deal of attention in the information recovery literature for seeking relevant information [9]. Exploring the contents of the web pages for automatic indexing is of fundamental importance for various web applications. The purpose of storing an index is to optimize speed and performance in finding relevant documents for a search query. Without an index, the search engine would scan every document in the corpus, which would require considerable time and computing power. For example, while an index of 10,000 documents can be queried within milliseconds, a sequential scan of every word in documents is a time consuming task. The additional computer storage required to store the index, as well as the considerable increase in the time required for an update to take place, are traded off for the time saved during information retrieval.

In AVL (i.e. height balanced binary tree) tree [3], the height of a tree is defined as the length of the longest path from the root node of the tree to one of its leaf node, and the balance factor (BF) is: (height of left subtree – height of right subtree). To call the AVL as balanced the value of BF should be -1, 0 or 1. This strategy makes the searching task faster and optimized.

2. 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 but they seem to be less efficient in efficiently accessing the index.

F. Silvestri, R.Perego and Orlando [4] proposed the reordering algorithm which partitions the set of documents into k ordered clusters on the basis of similarity measure where 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.

Oren Zamir and Oren Etzioni., [5] proposed threshold based clustering algorithm. Initially, the number of clusters is unknown, based on the specified threshold value the similarity between the two documents is evaluated and they are classified to the same cluster. 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.

C. Zhou, W. Ding and Na Yang [6], the paper introduces a double indexing mechanism for search engines based on campus Net. The CNSE consists of crawl machine, Chinese automatic segmentation, index and search machine. 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.

N. Chauhan and A. K. Sharma [7] proposed, the context driven focused crawler (CDFC) that searches and downloads only highly relevant web pages, thus, reducing the network traffic. A category tree has been used, which provides flexibility to the user for interacting with the system showing the broad categories of the topics on the web. The proposed

design significantly reduces the storage space at the search engine side.

P. Gupta and A. K. Sharma [8], worked on context based indexing in search engines using ontology. 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 critical look at the available literature reveals that there is a requirement for a technique to organize the keyword and their contexts in a better fashion as storing in a linear fashion makes searching of a document a bit time consuming.

3. PROPOSED WORK

This paper proposes an algorithm for indexing the keyword extracted from the web documents along with their context. The indexing technique uses a height balanced binary search (AVL) tree, in addition to improved performance in the retrieval of information, this data structure is able to support dynamic indexing, which is especially important for environments where documents are changed frequently. If the planning about the arrangement of the keywords is done then AVL tree can be achieved. The need to develop a technique that maintains a balance between the height of the left and right sub-trees of binary tree arises for the faster search of documents stored in the database according the their keywords.

Architecture of context based indexing is represented in Figure1. The web pages are stored in the crawled web page repository. Each web document is identified by its document-id. The preprocessing steps are performed on the documents (i.e. stemming as well as removal of stop words). The frequently occurring keywords are extracted from the document, and their corresponding multiple contexts are identified from the thesaurus. Indexer maintains the index of the keyword using the AVL 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.

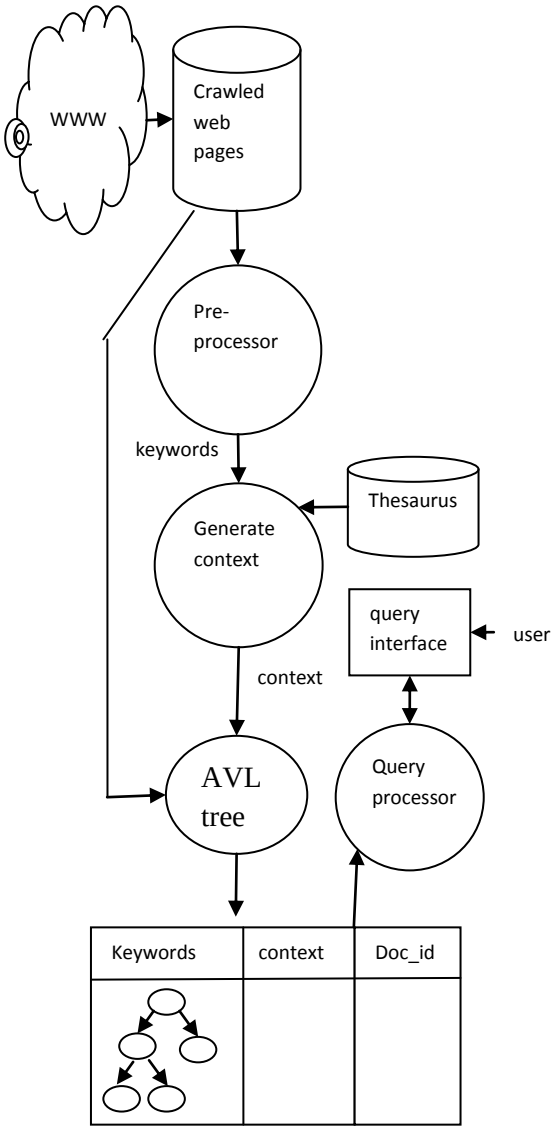

Figure 1: Architecture of context based indexing.

Figure 2 depicts the Context Based Retrieval Interface. The user enters the desired keyword (say Current), on the click of show context, the different contextual meanings of the keywords are displayed. The user selects the desired context of the keyword, through the show document button, the corresponding related web page URLs are listed (available in the repository) which can help the user to directly access more related and relevant information. This technique provides a fast access to document context and structure along with an optimized searching.

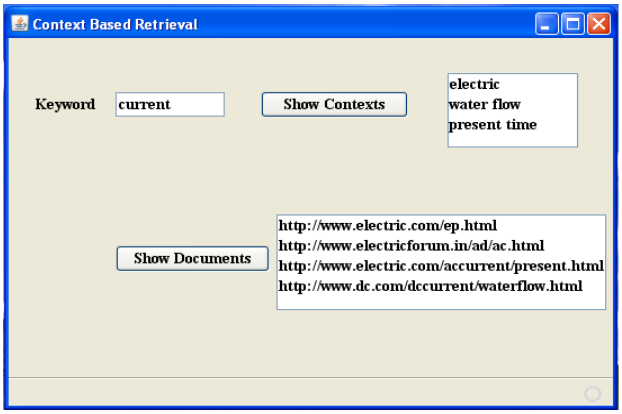

Figure 2: Context Based Retrieval Interface

3.1 Proposed algorithm for the indexing scheme

The algorithm depicted in figure 3 shows the various steps in the construction of the context based index and hence context based searching**.**

Step1: Preprocess the crawled web documents and extract the keyword along with their frequency of occurrence.
Step2: Input the keywords to the context generator which extracts the multiple contextual sense of the word. Context is being searched in the thesaurus (a dictionary of words available on WWW from thesaurus.com, which contains the words as well their multiple meanings).
Step3: The keywords along with the context are indexed using the AVL tree.
Step4: Compare the entered keyword with the node's keyword field of the AVL tree, until a similar word is found.
Step5: If search is not a success, create a node containing the following fields (Leftchild, keyword, rightchild, link) as shown in figure4.The link is pointer variable which points to the database where the context of keyword and the corresponding document_id is stored. Context is being searched in the thesaurus (a dictionary of words available on WWW from thesaurus.com, which contains the words as well their multiple meanings).
Step6: Arrange the node in the AVL tree, according to the height BF.
Step7: Repeat step 4, 5 and 6 until all the extracted keywords are arranged.
Step8: 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.
Step9. Thus, AVL indexing technique provides a fast access to document context and structure.

Figure 3: Steps involved in the construction of the context based index using AVL.

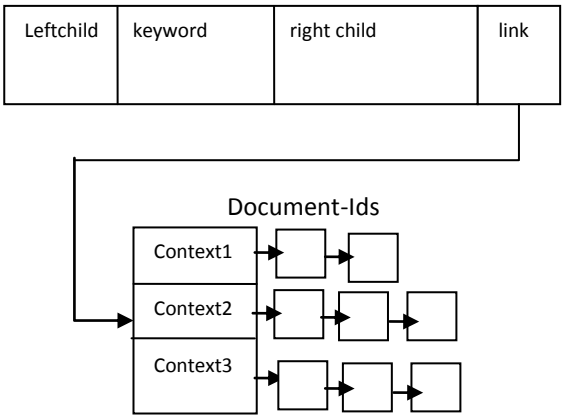

Figure 4: Node structure.

3.2 Steps are required to construct AVL tree

(i) Creation of binary search tree: This module creates a node corresponding to the various keywords and adds the related information in the node and finds suitable location in the binary tree.

```
Create_BST()        //initially the tree is empty.
{
   create new node containing the fields ( left child,
   keyword, rightchild, link).
  Leftchild value = NULL
  Rightchild value = NULL
   Link = address of database where the     context
   and the corresponding document_id is stored
  Insert_node();
}
Insert_node()
{
    Check, whether value in current node and a new
     keyword value are equal.
    If so, duplicate is found.
   Otherwise,
            if a new keyword value is less, than the
            root node's value:
                if a current node has no left child,
                     place for insertion has been
                  found;
                        otherwise, handle the left child
                        with the same algorithm.
                        Compute_height();
            if a new value is greater, than the root
            node's value:
            if a current node has no right child, place
            for insertion has been found;
                        otherwise, handle the right
                        child with the same algorithm.
                        Compute_height();
}
```

(ii) Insertion of a node may cause imbalance in a subtree and ultimately leading to imbalance of the whole tree. To call the AVL as balanced the value of BF should be -1, 0 or 1.

```
Compute_height()
  {
    If child = NULL then height is 0,
    The height of tree = 1+ max (height of left
                                 subtree, height of right
                                subtree)
          }
```

(iii) The rearrangement of the node can eliminate the imbalance.

```
Arrange_tree()
  {
Case1: Insertion of a node in the left subtree of the
  left child of the pivot.
  Perform rotation toward right of pivot.
Case2: Insertion of a node in the right subtree of
  the right child of the pivot.
  Perform rotation toward left of pivot.
Case3: Insertion of a node in the right subtree of
  the left child of the pivot.
  Perform left-right rotation from pivot.
Case4: Insertion of a node in the left subtree of the
  right child of the pivot.
  Perform right left rotation from pivot
  }
```

4. EXAMPLE

Consider the set of keywords (Table 1) extracted from the various web documents with Document_ID (1 – 9).

Table 1: List of keywords extracted from the various web documents

Keywords	Document_ID
Crawler	1,2,3,8
Apple	1,4,5
Repository	1,2,3,9
Current	1,2,3,5, 6, 7,9
Category	2,3
Ant	2,3,5
Account	1,2,3,5
Agent	1,2,3,5,6
Automatic	1,2,5,6

The data given in Table 1 would be represented by a BST as shown in figure 5.

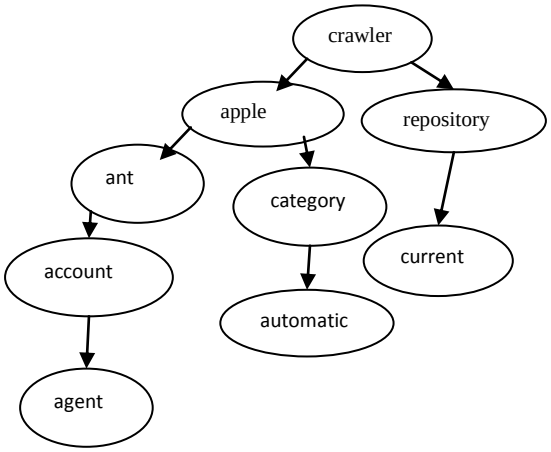

Figure 5: Representation of keywords using binary search tree

However, the proposed AVL tree indexing will represent the given keywords as shown in figure 6.

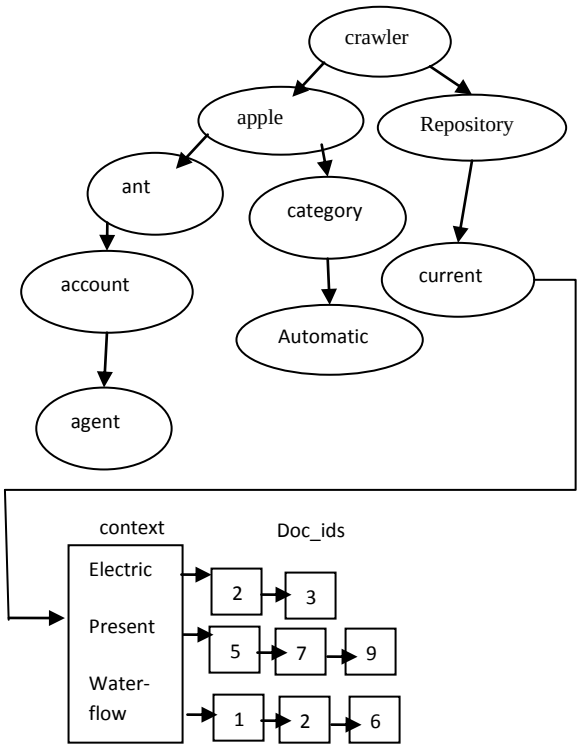

Figure 6: Representation of keywords using AVL tree along with context and doc-id.

The representation of keywords as organized in the figure 5 and figure 6 reveals the proposed representation not only increases the search efficiency but each node of the tree stores the contextual words of each keyword, leading to assignment of more relevance to the corresponding documents.

5. CONCLUSION

This paper proposes a technique for indexing the keyword extracted from the web documents along with their context. The AVL 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.

6. REFERENCES

[1] Diligenti M., Coetzee F.M., Lawrence S., Giles C.L.and Gori M., "Focused Crawling using context graphs", Proc. International Conference on Very Large Databases (VLDB '00), pp. 527-534, 2000.

[2] Yang Yongsheng and Wang Hui, "Implementation of Focused Crawler", COMP630D Course Project Report.

[3] A.K.Sharma, "Data Structures using C", Pearson publication, 2011.

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

[5] Oren Zamir and Oren Etzioni "Web Document Clustering: A feasibility demonstration". Proceedings of SIGIR, 1998.

[6] 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.

[7] Naresh Chauhan and A. K. Sharma," Design of an Agent Based Context Driven Focused Crawler",BVICAM'S International Journal of Information Technology, pp 61-66, 2008.

[8] 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.

[9] Steve Lawrence, "Context in Web Search", IEEE Data Engineering Bulletin, 2000.

[10] Wang Jicheng, Huang Yuan, Wu Gangshan and Zhang Fuyan, "Web Mining: Knowledge Discovery on the Web", IEEE International Conference, Tokyo, 1999.

[11] O. Zamir, O. Etzioni, O. Madanim, and R.M. Karp "Fast and Intuitive Clustering of Web Documents," Proceeding Third International Conference Knowledge Discovery and Data Mining, pp. 287-290, Aug. 1997.