Improving Data Access Performance by Reverse Indexing

Mary Posonia [#1], V.L.Jyothi [*2]
[#]Department of Computer Science, Sathyabama University, Chennai, India
[#]Department of Computer Science, Jeppiaar Engineering College
Chennai, India
[1]soniadelicate@gmail.com
[2]Jyothivl15@yahoo.com

Abstract— XML keyword search is exceptionally easy to understand strategy in the late patterns for the retrieval of the XML content. The unstructured XML substance is generally utilized with no issues. Significant trouble is to seeking in the structured and semi-structured data. Numerous algorithms are existing for searching the keyword and ranking based on some specific criteria. The search issue of retrieving the pointless record is not solved. Ranking is not based on word count which creates confusion for the user. In this proposed technique client even has an advantage of checking the outcomes by Reverse Indexing Technique with the ranked results to avoid redundancy, confusion, overlapping of documents and to achieve data integrity across diverse applications.

Keyword- XML, Indexing, Keyword search, Ranking

I. INTRODUCTION

This document is a To define the results of XML keyword search many valuable models are proposed and the most popular ones are the Smallest Lowest Common Ancestor (SLCA) [1][8] model and its variants, all of which are called as the LCA-based models. Many works mention that former LCA-based result models have false positive or false negative problems. Interestingly, their remedy approaches conflict with each other and counter examples can always be found to testify that the two problems still happen. The other models like MLCA (Multi-way Lowest Common Ancestor) and GDMCT (Grouped Distance Minimum Connected Tree) also generates the same answers like the SLCA based methods, as the both models exhibit same results similar to SLCA. Filtering is performed for such type of nodes and the left ELCA nodes are called as Valuable Lowest Common Ancestor (VLCA) nodes. The rule of VLCA method is actually kind of over strict and more relaxed criteria could be used such as the LCA have to be low or the compactness should be high. Some of the most useful information is omitted in their search results. Actually all the LCA-based models only serve the best results (in which all the specified keywords have to be contained) and refuse to organize and return the second-best ones.

So, the proposed system is introduced to overcome the aforementioned defects. This adaptive system containing the Partitioning of XML documents and Efficient Boolean Retrieval algorithm combines with the Ranking algorithm to perform in its best way. This system gives the high ranked XML documents within less time. Searching of the words in this system becomes easy when compared to the other systems.

II. RELATED WORK

After briefly reviewing the most important LCA-based result models [1][8] for XML keyword search, like SLCA, ELCA, VLCA, GDMCT, MLCA, it is known that all of these works model the underlying XML data as a rooted, labelled, unordered tree. The most important one of them is SLCA. In SLCA model a result is defined as a sub-tree that: (1) the labels of whose nodes contain all the keywords, (2) none of its sub-tree satisfies the first condition except itself. The root of such a sub tree is called a SLCA node. It's recognized that SLCA model is definitely not a perfect one. All the ELCA nodes can be retrieved through a straightforward two-step process: (1) find all the SLCA nodes, halt the process if there isn't any; (2) remove all the SLCA nodes along with the sub-trees rooted in them, then turn to the first step. It can be easily proved that the union of all the SLCA nodes obtained each time in the first step is indeed the set of ELCA nodes. Many other LCA-based models are also existing which are not so important because they are not much better than the existing systems. In addition, in other tree data model, XSeek [7] implements semantic inference to improve the results through the adoption of entities and keyword match pattern. It also works without schema information.

Ranking schemes have been studied for keyword search on XML documents. XRank [2] employs such a rank model that extends Google's Page-Rank to XML element level to rank all LCA results, which takes into account result specificity, keyword proximity and hyperlink awareness together. In addition, extended TF-IDF techniques stemming from traditional information retrieval have been adopted in many ranking schemes. However, TF-IDF [1] based ranking models can easily ignore the inherent hierarchical structural information of XML data, which may always indicate some important semantics. RACE is a ranking mechanism to rank

compact connected trees, by taking into consideration both the structural similarity and the textual similarity, thus leverage the efficiency and effectiveness of keyword proximity search over XML documents. In XSEarch, the XSEarch [14] Ranker ranks the results by giving a score to each result, considering both its structure and its content. eXtract system can generate self-contained result snippets within a given size bound which effectively summarize the query results and differentiate them from one another, according to which users can quickly assess the relevance of the query results. XReal adopts a novel idea to implement XML keyword search with relevance oriented ranking [13] . it develops a formulae to identify the search for nodes and searches via nodes of a query, and then adopts a novel XML TF*IDF ranking strategy, which basically utilizes the statistics of underlying XML data, to rank the individual matches of all possible search intentions.

III. PROPOSED SYSTEM

The Ranking of XML documents with adaptive system has been handled in this proposed system. A ranking is a relationship between a set of items such that, for any two items, the first is either 'ranked higher than', 'ranked lower than' or 'ranked equal to' the second. The XML documents are retrieved from the internet as XML benchmark datasets. These benchmark datasets are used for searching and ranking of XML documents within themselves. The user searches a keyword in the existing benchmark dataset to find his intended search. The XML documents retrieved as benchmark data sets are partitioned into number of words to make the search mechanism easy and implementing the Efficient Boolean Retrieval algorithm to form the entire process into an Adaptive system.

Major contribution to this paper:

- Distinct query inputs for both the XML tags and XML data to make it convenient for the user.
- Retrieving both the word count and total count of a particular document for the given search word.
- Saving time in retrieving the exact search word without any prefixes and suffixes for that word if existing in the document.
- Ranking of the searched documents based on the word countto make the work easier for the user.
- Matching of the ranked results with the results of Reverse Indexing method to confirm that there is no overlapping of documents and the data, which acts as a verification technique.

A. System Architecture

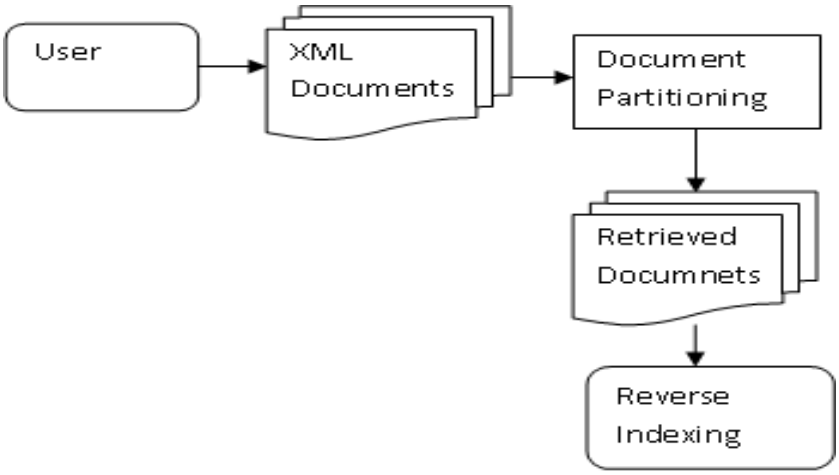

Fig .1. System Overview

B. Partitioning Method

The partitioning method used for the pre-processing of XML documents is the partitioning method. The XML benchmark data sets are collected and they are partitioned depending to the words to make the search easy. XML benchmark datasets used for the retrieval of keywords from them. All the content in the XML documents is partitioned into many nodes with help of the tokens

Algorithm 1: Partitioning XML Documents

```
Input : XML benchmark datasets
Output : XML Documents partitioned into nodes.
    Procedure :
    //data structure to keep files
    private Scanner input=new Scanner(System.in);
    private int token;  //token to identify the query
    private int matches[];  //tokens to identify matching events
    if(token==2 && found==1){  //assigning the suitable values for the tokens
     result=1; for(int x:matches)
   result=result*x; }else if(token==1 && found==1){
   result=0;  for(int x:matches)  result=result+x;
  }else if(token==0 && found==1) {result=0;
   for(int x:matches) result=result+x; }
```

Algorithm1 explains the concept of data partitioning in Xml documents. First select the path of the XML benchmark data sets in the hard drive. All the documents in that particular folder will be included for the partitioning of data. All the words in each document are partitioned into many nodes by considering each node as the token. These nodes which are partitioned are given as input for the searching algorithm.

C. Reverse Indexing

A reverse index is a term used to describe the practice of reversing index values in a database management system before they are indexed. This process is specifically useful in indexing and accessing archived information that is organized in a sequential manner. Within transaction processing systems that have a high volume of data transmission, reverse indexing allows for database management systems to operate more quickly and efficiently.

But here in this proposed system, as we are retrieving the links from the ranking module and storing into the database, it is difficult to reverse the key value and print it again in the database which creates confusion to the user who is referring to the database. So, when the links are stored into the database, these documents which is having the keywords has to be printed in a window to check whether any overlapping of documents is occurring after the ranking of the existing XML documents.

Algorithm 2: Procedure for Reverse Indexing

```
Input :  Ranked XML documents with a particular search word containing in it.
Output :  Collection of the links printed to avoid overlapping of the ranked files.
Process :
public void add(File file){  //adding files to the data structure
temp =new File[size];  //creat a temporary storing array
public File get(int index){  //getting out the files
 if(index<=size)  //check the index
 return bin[index];  else   return null;
 File inputFile = new File(jList2.getSelectedValue().toString());
   in = new FileInputStream(inputFile);
   byte bt[] =  new byte[(int)inputFile.length()];
   in.read(bt);  text = new String(bt);
   String word = jTextField2.getText(); int totalCount = 0;
   Scanner s = new Scanner(text); while (s.hasNext()) {
    totalCount++;  if (s.next().equalsIgnoreCase(word)) wordCount1++;
} jLabel6.setText(String.valueOf(wordCount1));
```

In reverse indexing all the files obtained from the Ranking process which uses search node algorithm. Print the files in which the search word given by the user exists out of all the existing documents in the XML benchmark datasets. Match the results of the ranking process with the files obtained by the printing of existing search word containing files. There should be a perfect match between these two sets of files, and it should not lead to any overlapping of files in the ranked results.

IV. RESULTS AND DISCUSSION

The results obtained in the proposed system for the work done are displayed in this section and they are discussed accordingly to give the clear idea to the users or validators.

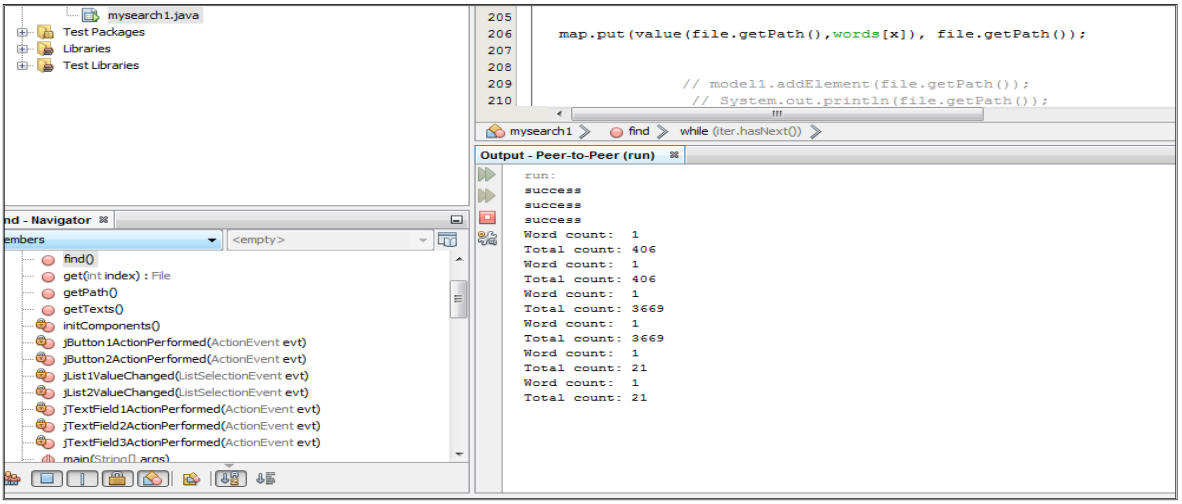

Fig. 2. Word Count

Partitions the XML documents into many numbers of words by assigning them as tokens. These tokens are easily matched with search word in all the existing documents and word count is obtained for each and every document in which particular search word is existing. In the figure 2, both the total count and word count for every document are displayed for every possible document.

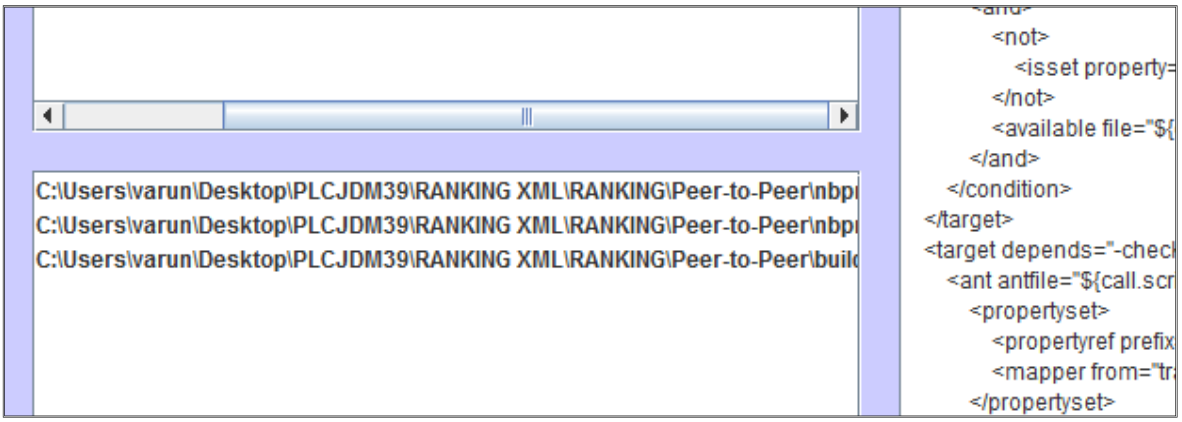

Fig .3. Reverse Indexing

Reverse indexing technique in this proposed system is to print the results of ranking process in the reverse order. After printing them, the user has to check the results and compare them to find whether there is any overlapping of files after obtaining of the results. The figure 3 gives the idea how to compare the results in both the ranking and reverse indexing list boxes.

A. Performance Analysis

$$\text{Top} - \text{k Precision} = \left(\frac{\text{A}}{\text{A} + \text{B}}\right) * \ 100\%$$

where A: Number of relevant records retrieved

B: Number of relevant records not retrieved

C: Number of irrelevant records retrieved

TABLE 1. Finding similarity

XML Dataset	Word-count	Average-no of Similar Key-tags
DBLP	406	60%
SIGMIOD-RECORD	3669	75%

TABLE 2. File Retrieval

Total size of File	Word corrected in no.of places by Compiler	No.of Relevance File Retrieved
1MB	3	2
5 MB	1	5
10 MB	4	9

V. CONCLUSION

The Ranking of XML documents with adaptive system has been taken care of in this proposed framework. The client searches a keyword in the current benchmark dataset to discover his intended search. The XML documents retrieved as benchmark information are partitioned into number of words to make the search mechanism simple. At the point when the search is completed on the premise of word count, these documents are ranked by the maximum number of search words in it, utilizing the search node technique to place them in descending order of the word count. Such content obtained is stored into the database for future inquiry of the information. These reports are transformed in a system called reverse indexing to enhance the data integrity of the framework and to utilize it across diverse applications.

REFERENCES

[1] Papakonstantinou. Y and Xu. Y, “Efficient Keyword Search for Smallest LCAs in XML Databases,” in Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data (SIGMOD’05), 2005, pp. 537–538.
[2] Botev. C, Guo. L, Shanmugasundaram. J and Shao. F, “XRANK: Ranked Keyword Search over XML Documents,” in Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (SIGMOD’03), 2003, pp. 16–27.
[3] Hristidis. V, Koudas. N, Papakonstantinou. Y, and Srivastava. D, “Keyword Proximity Search in XML Trees,” IEEE Trans. Knowl. Data Eng. (TKDE), vol. 18, 2006, issue no. 4, pp. 525–539.
[4] Gilleron. R, Kong. L and Lemay. A, “Retrieving Meaningful Relaxed Tightest Fragments for XML Keyword Search,” in Proc. 2009 Inter- national Conference on Extended Data Base Technology (EDBT’09), 2009, pp. 815–826.
[5] Feng. J, Li. G, Wang. J and Zhou. L, “Effective Keyword Search for Valuable LCAs over XML Documents,” in CIKM, 2007, pp. 31–40.
[6] Jagadish. H, Li. Y and Yu. C, “Schema-Free XQuery,” in Proceedings of the 30th International Conference on Very Large Data Bases (VLDB’04), 2004, pp. 72–83.
[7] Chen. Y, Liu. Z and Walker. J, “XSeek: A Semantic XML Search Engine Using Keywords,” in Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB’07), 2007, pp. 1330– 1333.
[8] Chan. C, Goenka. A and Sun. C, “Multiway SLCA-basedKeyword Search in XML Data,” in WWW, 2007, pp. 1043–1052.
[9] Papakonstantinou. Y and Xu. Y, “Efficient LCA Based Keyword Search in XML Data,” in Proc. 2008 International Conference on Extended Data Base Technology (EDBT’08), 2008, pp. 535–546.
[10] Chen. Y and Liu. Z, “Identifying Meaningful Return Information for XML Keyword Search,” in Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data (SIGMOD’07), 2007, pp. 329–340.
[11] Li. N, Yang. W, Zhu. G, and Zhu. H, “Adaptive and Effective Keyword Search for XML,” in PAKDD, 2011
[12] Yang. W and Zhu. H, “Semantic-Distance Based Clustering for XML Keyword Search,” in PAKDD, 2010, pp. 398–409.
[13] Bhawani Shankar Burdak and Laxmi Choudhary, “Role of Ranking Algorithms for Information Retrieval,” in Cornell University Library-Computer Science, 2012.
[14] Cohen. S, Kanza. Y, Mamou. J and Sagiv. Y, “XSEarch: A Semantic Search Engine for XML,” in Proceedings of the 29th International Conference on Very Large Data Bases (VLDB’03), 2003, pp. 1069– 1072.