A Search Algorithm Oriented to XML Keywords

Xingyuan Li

College of Information Engineering, Ningbo Dahongying University.
Ningbo, zhejiang, China

Abstract

It is one of the core issues of XML keywords search algorithm about putting the hierarchical structure information of XML data into the index and making it support the efficient keyword search algorithm. This paper proposed a new a retrieval algorithm which was based on LAF coding; HBA, a bottom-up XML keyword search algorithm, and it can support a variety of search semantic models effectively. In experimental comparison with traditional keywords search algorithm, HBA algorithm not only has advantage of greater time efficiency, but also supports various XML keywords search semantic models effectively.

Keywords: *XML keywords search, HBA algorithm, Two-layer, LAF numbering.*

1. Introduction

[1]The appearance of XML is a very important milestone in the history of the development of the Internet. With the maturation of the XML application, a large number of XML documents are used for data transmitting, data storing and data exchanging. Hence, effective searching and user's the data using have become major challenges of internet. At the same time, XML search has great meaning to the realization of semantic search.

The biggest difference between XML data and plain text is that XML data has structure information besides content. This helps XML to describe data more accurately. How to code the hierarchy information into indices for supporting highly efficient keyword search algorithm is one of the core issues in the field of XML keyword search.

Researchers in the field of information retrieval have paid wide attention to the XML keywords search, and some domestic researchers also achieved certain results about the XML keyword search in recent years. It is well known that Dewey coding is a more popular XML elements coding method, therefore, many search algorithm was proposed based on Dewey coding.[2] Firstly encoded XML elements by Dewey coding; [3] solved the XML research result problem based on the Dewey encoding stack-based

Algorithm and the stack-based algorithm could solve the ELCA effectively through the ELCA model as a query semantic model; [6-7] proposed a new retrieval semantic model SLCA, but also solved SLCA and Scan Eager algorithm and Look-up Eager algorithm. The SLCA model could meet the basic needs of the user's query, and it is a more popular result searching model. [4] Proposed to use the LISA algorithms to solve the SLCA results, certainly, LISA adopts a bottom-up query strategy.

Dewey number is an effective method to code hierarchical information. [5]Many algorithms based on Dewey number are proposed. However, Dewey number has two obvious shortcomings. Firstly, the length of the Dewey number for an XML element increases with the depth of this element in a XML tree, which may cause indexing redundancy when processing large scale documents set; secondly, many algorithms based on Dewey numbers need to sort elements according to the lexicographic order of Dewey numbers, the complexity for comparing two Dewey numbers is O (N) (here N is average length of Dewey numbers), which will be unacceptable in processing large scale XML documents set.

XML keywords search algorithms are mostly based on the Dewey encoding. However, these algorithms' time performance is not very high when dealing with large-scale set of XML documents. This paper proposes a new a retrieval algorithm which was based on LAF coding; HBA, a bottom-up XML keyword search algorithm, and it can support a variety of search semantic model effectively. Through the experiment, the proposed algorithm obtains greater improvement in space efficiency and time efficiency comparing to the traditional approach.

2. Basic concepts

2.1 Search semantic model

[8]Search semantic model is the basis of XML keywords search, and it is denned as follows.

Definition 1: SLCA(Smallest Lowest Common Ancestor): define inquiry Q={k1,k2 ⋯ kn}, and k1,k2 ⋯ kn are the different keywords corresponds to the different nodes in DOM tree such as S1， S2⋯Sn which was including the keywords ki(0<i <= n), so,{LCA(S1, S2 ⋯ Sn) | N ∈

LCA(S1,S2 ⋯ Sn) → ∃ Ni ∈ Si(0<i<=n) ∧ N=LCA(N1,N2...Nn)}and define SLCA as follows: {SLCA(S1, S2⋯Sn) | N∈SLCA(S1, S2⋯Sn)→ N ∈ LCA(S1, S2 ⋯ Sn) ∧ (N ∈ SLCA(S1, S2 ⋯ Sn) → (ancestor(N,N)→N=N))}. The function ancestor (N, N) is used to determine the inheritance relationship of two nodes N and N, if existence, the function would return true, else, return false. This paper mainly uses SLCA model as the search results model. Stack algorithm, Scan Eager algorithm and Look-up Eager algorithms are the most frequently cited in several search algorithms, and this paper is going to compare the efficiency of HBA with the other algorithms.

2.2 Encoding

Definition 2: stability of coding: according to some encoding methods, each element in an XML document can get its only coding. In turn, the encoding is stable if it can uniquely determine a stable structure of the XML document (XML tree) through the group encoding of XML elements.

Definition 3: the floor traversing of tree: while floor traversing a tree, we visit the node firstly whose depth is 1 (the root node), and then, visit the node whose depth is 2、3…n until all nodes in the tree visiting entirely. The floor traversing names a unique serial number for each node in XML tree. For instance, fig.1 the floor traversing results are A, B, C, D, E, F, G, H, I, J, and the floor traversing names an unique serial number such as 0,1,2,3, 4,5,6,7,8,9.

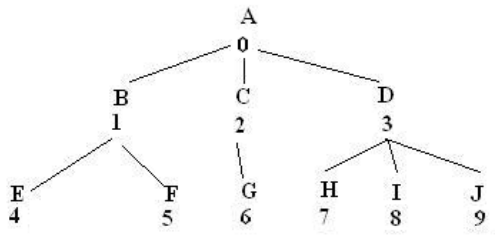

Fig.1 Tree level traversal

Definition 4: LAF coding: abbreviation of Level order And Father numbering, compared to Dewey coding, LAD coding is based on global coding strategy which is the floor traversing of tree (breadth-first traversal).For the XML DOM tree nodes, the LAF coding consists of three parts: the sequence number of the floor traversing node, the sequence number of the father node and the depth of the node. Due to the root node has no father node, the sequence number is set to -1.

LAF coding which is similar to Dewey coding may be stored by the vector with a length of 3 which presents three parts of the LAF coding respectively. And the three parts are separated by '.' for the ease of reading.

Fig.1 the XML document using the LAF encoding results. As for the root node, its floor traversing sequence number is 0 with the depth being 1, and its father floor traversing sequence number is -1, so it's LAF coding is 0.-1.1. The node, its floor traversing sequence number is 4 with the depth being 3 and its father node's traversing sequence number is 1, so its LAF coding is 4.1.3. The LAF coding of other nodes could be obtained similarly.

2.3 LAF coding properties

Property 1: it is obvious that the LAF encoding length is 3 for an arbitrary XML element.

Property 2: LAF coding is stable. Firstly, the LAF coding is different for any two elements of an XML element because the floor traversing sequence number of each element being different. Secondly, the XML tree is unique for a LAF coding set which could form a tree.

Property 3: For LAF coding, when comparing the size of LAF coding, they shall only be compared by the floor traversing sequence number whose time complexity is O (1).

Property 4: For LAF coding A, B of any two elements in the same document, if the floor traversing sequence number of A, LA is greater than the floor traversing sequence number of B, LB that is LA≥LB. Hence, FA≥FB and DA≥DB, FA , FB are the father floor traversing sequence numbers and DA , DB are the depth of A, B nodes. So, if LA≥LB, we can confirm that FA≥FB and DA≥DB. If DA≥DB, we can confirm that FA≥FB and LA≥LB.the character of floor traversing is the floor traversing sequence numbers would be more and greater by the increase of depth, so Property 4 could be confirmed.

Property 5: For LAF coding A, B of any two elements in a same document, the time complexity is O (N) when seeking their common ancestor LAF coding, N are the depth of A, B. When seeking common ancestor of A and B, the worst case is their common ancestor being the root node, this time, the time complexity of O (N).

3. HBA algorithm

3.1two floor index based on LAF coding

[5] Index is the basis of efficient XML search system. Two floor indexes can greatly increase the efficiency of the index system through storing the normal attributes and semi-structured text attributes of XML document separately.

Definition 5: two floor index: a new XML index structure, in the index structure, the first floor stores the normal attributes of XML document and the second floor stores semi-structured text attributes of XML document associate these two floor index by pointer. Inverted index consists of the first floor index and the second floor index. The storage information in first floor index includes document number, document links address, XML element floor traversing sequence number list and so on. The storage information in second floor index is the LAF list of every XML document.

Definition 6: LAF list: the ordered coding set of all the numbers from small to large according to the LAF coding in XML document, XML document corresponds to a unique LAF coding list.

3.2data structure

[6] Heap is a common data structure in sorting algorithm; heap sorting is an efficient sorting method. The substance of Heap is a binary tree, an important property of heap is that the heap could be the value of the binary tree roots, and it could also be the maximum or minimum value of all the binary tree nodes. Beside that, Heap could be the sub trees whose parent nodes are the root. HBA algorithm bases on the maximum Heap, and achieves the keyword searching result from top to bottom.

Definition 7: the maximum heap: A complete binary tree T which contains n nodes, its floor traversing result is K0, K1, and K2…Kn-1 , Ki meaning the weight or value of the i-th node in the floor traversing, so they have the following properties:

$$\begin{cases} K_i \geq K_{2i+1} \\ K_i \geq K_{2i+2} \end{cases} \left(i = 0,1 \ldots \left\lfloor \frac{n-1}{2} \right\rfloor \right) \quad (1)$$

The maximum heap has the following two basic properties:

(1) The maximum heap corresponds to a complete binary tree and the weight of the roots in all nodes (value) is maximum.

(2) The left and right sub trees are also the maximum heap whose parent node is the root node.

Fig.2 is an example of the maximum heap. Its floor traversing results in the complete binary tree is 100,90,75,33,89,36,53. Through the maximum heap definition, weight relationship between nodes in the binary tree conforms to the definition of maximum heap.

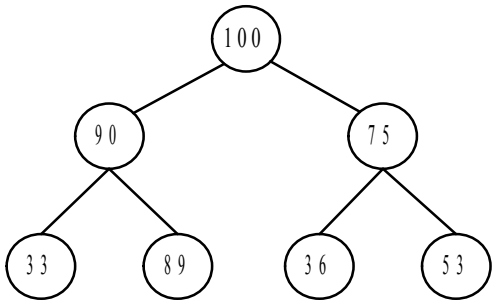

Fig.2 Maximum heap sample

3.3 description of HBA algorithm

HBA is short for Heap Based Algorithm on XML Keyword Search, its data structure is the maximum heap, and through processing the node of XML tree from bottom to top, it could obtain the keywords search result of XML efficiently.

HBA algorithm is based on two-floor index of LAF coding, and its core idea is: firstly, to find the XML document set which includes the keywords based on the first-floor index, then obtain the document set H, which includes all the keywords based on the intersection from the document set, secondly, based on the second LAF list, to obtain the search semantic model(for example SLCA) from every XML document in the document set H, at last, to supply the search result to the users. Hence, the HBA algorithm should include two parts:

(1) Obtain the XML document set, H, which includes all keywords.

(2) Obtain the XML elements which are suitable to the search semantic in every XML document in the document set H.

The first part of the algorithm is actually ordinary text search, because the ordinary text search does not regard to the internal structure of the document, the search efficiency can be very high. The reason why this part of the processing is needed is that in order to reduce the

processing invalid XML elements. The theoretical basis of the first part refers to the following necessary conditions.

The necessary condition that the XML elements being for search result: the necessary condition is that the XML element in the document set H, e, would include all the keywords in the inquiry q which was the XML keywords inquiry, that means the document H needed include all the keywords.

The above necessary condition determines the first portion of the algorithm could greatly reduce the ineffective treatment of the element.

From the above necessary condition, we know that the XML elements would not include all the keywords if a XML document, H, could not include all the keywords, so that XML element is not satisfied as the search result of the necessary conditions.

So, the mainly work of the first part of the algorithm is to exclude the XML document which did not contain all of the keywords, leaving only the XML document which contained all of the keywords. Therefore, the algorithm avoids the processing of XML elements that did not contain all of the keywords of the XML document, thereby accelerating the processing speed of the algorithm.

The second portion of the algorithm is the core of the HBA algorithm. The difference between the XML keywords search with conventional text search is to return more precisely search result to the users by the internal structure of the XML document.

XML search does not return an entire XML document but the internal XML element as the search results. The first part of the algorithm can be fully functional if the entire document was returned. The algorithm core idea of the second part is to find XML search results which were suitable to a specific semantic model from bottom to top by the maximum heap data structure and the LAF list.

The processing (for example, to solve the SLCA) is to obtain the document set, D, which included all the keywords through the first part of the algorithm.

As to every XML document, d, the number set which included all the node of keywords k would be obtained from the document d, and then, to obtain the union set, S, by the collection of the keywords number from the keywords inquiry, q, besides that, to mark the status of the node and construct the maximum heap H based on Sift the heap H was not empty, delete the root R of H. If R already was the node of SLCA, store the R condition to the father node F (from the LAF list) and insert F to heap H; or, if R already included all the keywords, output the R as the result, mark the R as the SLCA node, store the R condition to the father node F and insert F to the heap H. Sequentially process until the heap H is empty.

The code of the HBA algorithm shown in Fig.3.

In the algorithm showing in Figure 3, 1-8 rows are the first part of algorithm and 9-33 rows are the second part of algorithm. Through the earlier analysis of algorithms, it is obvious that the first portion of the algorithm is mainly processing according to first floor index of the LAF coding and the second portion of the algorithm is mainly processing according to the LAF list.

```
FUNCTION HBA(K1, K2,... Kn)
1   FOREACH keyword Ki
2       get its inverted document list Li from the first layer index;
3       IF Li is empty
4           printf("No results found");
5           exit;
6       END IF
7   END FOREACH
8   L0 = INTERSECTION (L1, L2, L3... Ln); //get the intersection of L1, L2, L3....Ln
according to document id;
9   FOREACH document Di in L0
10      Get the LAF table LAFi for Di from the second layer index;
11      Get all the level order number lists LO1, LO2... LOn for K1, K2,... Kn ;
12      Push the level order number in LO1, LO2... LOn into the max-heap and
initial their status;
```

```
13      WHILE max-heap is not empty
14          Get and delete the items with the same max level order number;
15          Combine the status of these items;
16          Get the LAF number from LAF_i through the max level order number;
17          Check the status of current LAF number;
18          If ( LAF number is SLCA already)
19              Get the father node's LAF number of current LAF number from
LAF_i;
20              Add the status of current node to its father node's status;
21              Push the father node's level order number into the max-heap;
22          ELSE
23              If current LAF number contains all the keywords
24                  Output current LAF number;   //Output SLCA results
25                  Marks current LAF number with SLCA status;
26              Add the status of current node to its father node's status;
27              Push the father node's level order number into the max-heap;
28              END IF
29          END IF
30      END WHILE
31  END FOREACH
```

Fig.3 HBA code

4. Experimental results and analysis

4.1 Experimental platform

Test the performance algorithm which proposed in the paper and compare to Scan Eager algorithm.

Experimental operating environment is: 1 IBM server as the service terminal, 1 PC computer as the client. The configuration of PC computer has multi-core E2160, 1.8GHZ, CPU, 2G memories and 160G hard disk. The configuration of IBM server has Intel Xeon E5405, 2.0GHZ, CPU, 8G memory and 1.5TB hard disk. The software environment is that the server's operating system being Window Server2003 and PC's operating system being Window XP. Programming Environment is Windows Studio2008.NET. The language of core program is C++ and the index database is MySQL5.0 in the paper.

The data set used in this experiment included four real data sets and one artificially generated data sets. The four real data sets are DBLP, Wikipedia, NASA and SIGMOD Record and the artificially generated data sets is XMARK. The five data sets show in Table 1.

Dataset	*Size (Mb)*	*Document number*	*Maximum depth*	*Elements*
DBLP	103	215493	5	3407057
NASA	22.4	2435	8	791177
SIGMOD	3.3	989	8	34572
WIKIPEDIA	66.7	3934	58	480091
XMARK	113	5751	13	2853356

Table 1: Data set attributes

4.2 Time Efficiency Comparison

The HBA algorithm in the paper based on the SLCA index semantic model puts the algorithm which could obtain SLCA result fatly based on the second floor index of LAF encoding. In the most tradition search algorithm based on SLCA, Scan Eager is mostly cited and also is proved to be the most efficient an algorithm. Then, comparison efficiency of the HBA and Scan Eager is followed.

This paper compares the efficiency of the HBA and Scan Eager by using 2, 3, 4 keywords in the different five data sets.

Fig.4 is the time efficiency comparison results of the two algorithms in SIGMOD data sets. From the comparison results, the HBA algorithm efficiency is significantly better than traditional Scan Eager algorithm efficiency. But there is little differences in efficiency of the two algorithms when four keywords. After analysis of the 10 group generated search keywords from the 4 keywords, it is found that the generated keywords all had low frequencies, most appearance only once in a XML document, this may lead to the result in the efficiency of the two algorithms become almost same.

Fig.5 is the time efficiency comparison results of the two algorithms in NASA data sets. From the comparison results, it is obvious that the HBA algorithm efficiency being significantly better than traditional Scan Eager algorithm efficiency.

Fig.6 is the time efficiency comparison results of the two algorithms in Wikipedia data set. From the comparison results, it is also obvious that the HBA algorithm efficiency being significantly better than traditional Scan Eager algorithm efficiency.

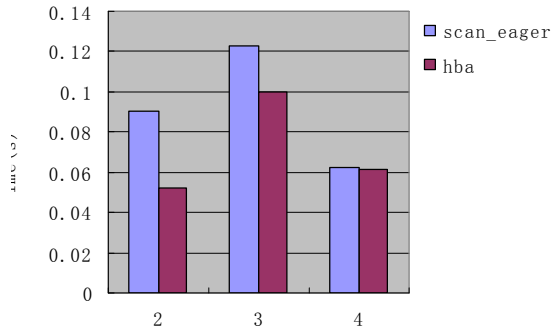

Fig.4 SIGMOD data set

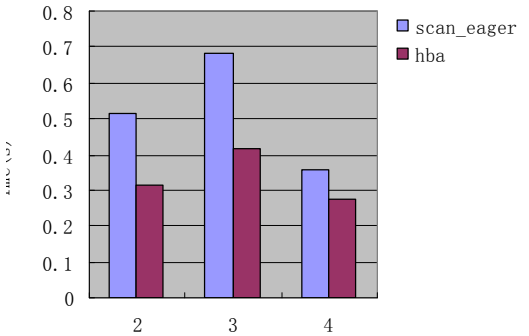

Fig.5 NASA data set

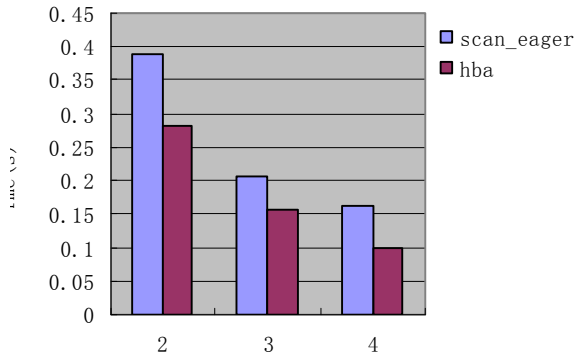

Fig.6 WIKIPEDIA data set

Fig.7 is the time efficiency comparison results of the two algorithms in DBLP data set. From the comparison results, the traditional algorithm is better when used 2 keywords, two algorithms is almost the same efficiency when used 3 keywords, but, HBA algorithm is better than Scan Eager algorithm when used 4 keywords. In this data set, HBA does not reflect the absolute advantage because the average depth of the elements in DBLP is only 2.4. The coding of LAF is longer than Dewey. But why HBA efficiency advantage becomes more and more obvious following the quantity of keywords increasing? That because the HBA might handle invalid element less with the increase of the keywords. It is obvious that the document number would be a corresponding reduction which contains all the certain keywords by the increase of keywords.

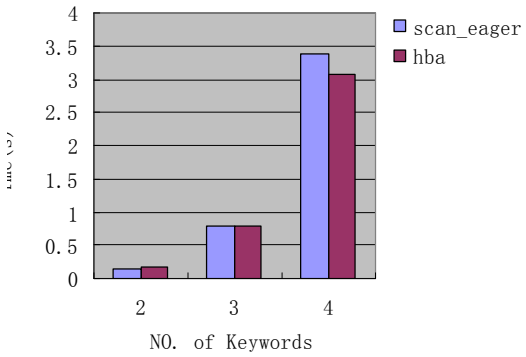

Fig.7 DBLP data set

Fig.8 is the time efficiency comparison results of the two algorithms in XMARK data set. From the comparison results, it is also obvious that the HBA algorithm efficiency is significantly better than traditional Scan Eager algorithm efficiency.

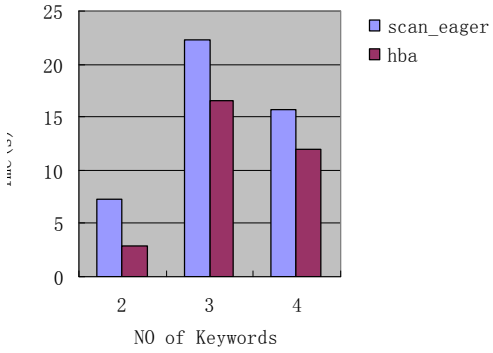

Fig.8 DBLP data set

From the above experiments, by using different data sets and the keywords, the time efficiency comparison between the HBA algorithm and Scan Eager algorithm, HBA algorithm seems superior to the traditional Scan Eager algorithm in most case. However, there are exceptions that the traditional algorithm is better than HBA algorithm when searching with 2 keywords. Generally speaking, the HBA algorithm is superior to the traditional Scan Eager algorithm when the keywords are more evenly distributed in the document set which average depth is higher (average depth is greater than 4).

5. Conclusion

To sum up, this paper presents the time efficiency assessment between a new index algorithm and other

search algorithms. It is verified by the experiments that the HBA algorithm efficiency is significantly better than traditional Scan Eager algorithm efficiency when the elements average depth is greater than 4 and the keywords are distributed evenly.

References

[1] Rada Chirkova;Leonid Libkin and Juan L. Reutter, " Tractable XML data exchange via relations ", Frontiers of Computer Science, Vol. 6, No. 3, 2012, pp. 243-263.

[2] Takahiro Komamizu; Toshiyuki Amagasa; Hiroyuki Kitagawa, "Faceted Navigation Framework for XML Data ", International Journal of Web Information Systems, Vol. 8, No. 4, 2012.

[3] Wu, Xiaoying;Souldatos, Stefanos;Theodoratos, Dimitri;Dalamagas, Theodore;Vassiliou, Yannis;Sellis, Timos, "Processing and Evaluating Partial Tree Pattern Queries on XML Data ", Knowledge and Data Engineering, IEEE Transactions on, 2012,Vol. 24, No. 12,pp.2244-2259.

[4] Meghdad Mirabi;Hamidah Ibrahim;Nur Izura Udzir;Ali Mamat, "An encoding scheme based on fractional number for querying and updating XML data ", Journal of Systems and Software, 2012,Vol.85, No. 8,pp.1831-1851.

[5] Nguyen, Khanh; Cao, Jinli, "Top-K data source selection for keyword queries over multiple XML data sources. " Journal of Information Science, 2012, Vol.38, No. 2, pp.156-175.

[6] Alfredo Cuzzocrea and Elisa Bertino, "Privacy Preserving OLAP over Distributed XML Data: A Theoretically-Sound Secure-Multiparty-Computation Approach ", Journal of Computer and System Sciences, 2011, Vol.77, No. 6, pp.965-987.

[7] Gilbert Tekli; Richard Chbeir; Jacques Fayolle, "XA2C: a framework for manipulating XML data ", International Journal of Web Information Systems, 2011, Vol.7, No. 3, pp.240-269.

[8] Nguyen, Khanh; Cao, Jinli, "Top- k answers for XML keyword queries ", World Wide Web, 2012, Vol.15, No. 5, pp.485-515.

[9] Bao, Zhifeng;Lu, Jiaheng;Ling, Tok Wang; hen, Bo, " Towards an Effective XML Keyword Search ", IEEE Transactions on Knowledge and Data Engineering, 2010, Vol.22, No. 8, pp.1077-1092.

[10] Xiping Liu;Changxuan Wan; Lei Chen " Returning Clustered Results for Keyword Search on XML Documents ", IEEE Transactions on Knowledge and Data Engineering, 2011, Vol.23, No. 12, pp.1811-1825.

Xingyuan Li male, 1979, College of Information Engineering, Ningbo Dahongying University.Xueyuan Road No.999, 315175 Ningbo, China. Graduated from Guilin University of Electronic Technology, master of published papers, core, presided over the municipal level above topics. The main research direction of information retrieval, data mining, computer network.