Adaptive Top-k Algorithm in SLCA-Based XML Keyword Search

Hang Yu, Zhihong Deng (Corresponding author), Yongqing Xiang, Ning Gao, Ming Zhang, Shiwei Tang

Key Laboratory of Machine Perception (Ministry of Education), School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China

pkucthh@gmail.com, zhdeng@cis.pku.edu.cn, xiangyq@cis.pku.edu.cn, nanacream@gmail.com, mzhang@net.pku.edu.cn, tsw@pku.edu.cn

***Abstract*— Computing top-k results matching XML queries is gaining importance due to the increasing of large XML repositories. In this paper, we propose a novel two-layer-based index construction and associated algorithms for efficiently computing top-k results for SLCA-based XML keyword search. We have conducted expensive experiments and the results show great advantage on efficiency compared with existing approaches.**

I. Introduction

In recent years, the ability to compute top-k answers to XML queries is gaining considerable attention due to the rapid growth of XML repositories. Top-k query evaluation on exact answers is appropriate when the answers are large and users are only interested in the highest-quality matches. Some works have taken efforts to resolve top-k applied in XML keyword search, such as Top-X system [1], [2] and XRank system [3]. However, top-X could not construct sub-tree automatically without structural information, but merely returns individual keyword nodes that contains the corresponding keywords instead of sub-tree results. So, it is very difficult for top-X to find top-k results that are defined by SLCA. In addition, because XRank organizes the inverted lists at the granularity of elements, it has to apply more random access to the inverted lists to get the corresponding nearest nodes to generate the sub-tree results [3].

In this paper, we propose a novel two-layer-based index construction and associated algorithms for efficiently computing top-k results for SLCA-based XML keyword search. The remainder of the paper is organized as follows. Section II mainly talks about some preliminaries such as query semantic and corresponding score function for XML sub-tree-based results. In Section III, we illustrate the approach for index construction (TLI), and associated algorithm (STA) will be proposed In Section IV. Our experimental results are showed in Section V. Section VI draws the conclusion and discuss future work.

II. Preliminaries

In this part, we mainly talk about two issues as preliminaries for our work. They are query semantic and score function.

In recent years, many query semantics have been proposed, such as XRank [3], Smallest LCA (SLCA) [4], Meaningful LCA (MLCA) [5], Grouped Distance Minimum Connecting Tree (GDMCT) [6], XSeek [7] and so on. In the paper, we employ SLCA, which is widely accepted, as query semantic. The result of SLCA model must satisfy two restrictions: (i) The results of SLCA should contain all keywords either in their labels or in the labels of their descendant nodes and (ii) they have no descendant node that also contains all keywords.

The critical issue of score function applied in XML documents is how to take the inner structure of XML tree into consideration. The score function should reflect both content information and structural information to get an overall score for each term and its corresponding ancestor tags. In our realization, we borrow the score function which is applied in Top-X system [1], [2]. Here we will not bother to talk about its principle and realization any more. In the following parts, we will simply use ***S(tag, term)*** to denote score computed for a pair of tag and term, and our index construction and system algorithm will also be stated based on pre-computed scores.

III. Index Construction

In this part we propose our TLI (two-layer-based inverted) index design for indexing XML documents. We use XML-specific extension to Okapi BM25 [1] as our score function to get the corresponding tag-term score. The aim of the index construction should not only contain the structural information of the XML documents but also support efficiently obtaining top-k query results according to associated query algorithm. Motivated by this thought, we extend the classical per-term inverted index to a two-layer-based inverted index construction for XML keyword search, as shown in Fig. 1:

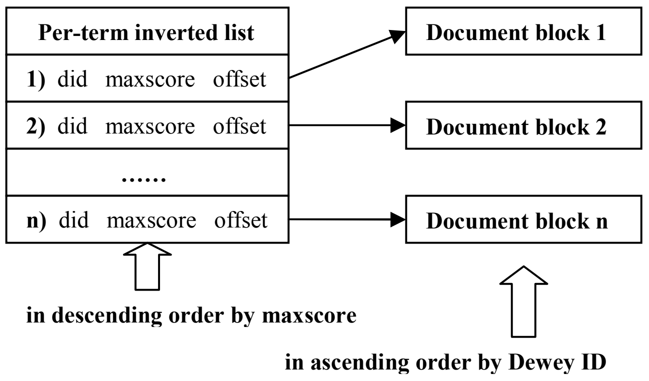

Fig. 1 Two-layer-based invert index construction

The left part in Fig. 1 is the first level of index construction, which is the same as the classical per-term inverted index with keywords as index entry. Each item in the inverted list is a table containing three elements (*did, maxscore, offset*), in which *did* is the identifier of XML documents in data set, and *maxscore* is the highest score in the corresponding *Document block* which we will illustrate later, and *offset* is an integer to tag the starting position of its corresponding *Document block* as shown in the right part in Fig. 1. The inverted list is organized in descending order with regard to *maxscore*.

The *Document block* is the second level of index construction. Generally each *Document block* is corresponding to document *did* and it is also an inverted list in ascending order with regard to Dewey ID. Each item in the inverted list of *Document block* contains two elements and each item is associated with a linked list that records the scores of all its ancestors. The following Fig. 2 is an example of *Document block* for keyword "forward".

DeweyID	**text**	**l inked list for all ancestor tags**			
		name	player	players	team
0.1.0.1.0	forward	0.1.0.1	0.1.0	0.1	0
		0.5	0.4	0.3	0.1
		name	player	players	team
0.1.1.1.0	forward	0.1.1.1	0.1.1	0.1	0
		0.5	0.4	0.3	0.1

Fig. 2 Example of Document Block for keyword "forward"

To calculate the *maxscore* of the *Document block*, we just need to get the maximal score of all scores in the linked lists.

IV. Algorithm

This section presents the core SLCA-based threshold algorithm (STA). Our basic idea for achieving top-k results in SLCA-based XML keyword search is employing the classical TA family of algorithms for candidate pruning and adaptive scheduling decision. However we make several changes across the process of algorithms to adapt it to SLCA-based XML keyword search. Fig. 3 shows the algorithm architecture and data flow on processing.

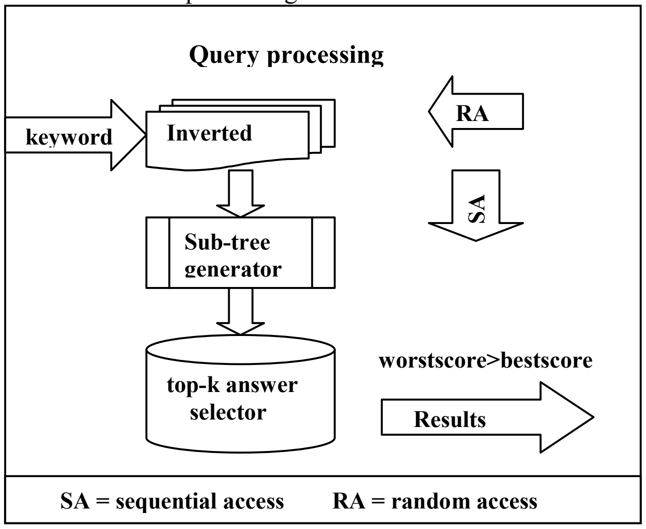

Fig. 3 Algorithm architecture and data flow

The input of the algorithm is a list of keywords and the output is sub-tree results associated with the query keywords. The algorithm process is constituted of three steps: 1) access to keyword inverted lists 2) the sub-tree generator 3) top-k answers selector.

1) Access to keyword inverted lists: Our query processing method is based on pre-computed two-layer-based inverted index and the first level is sorted in descending order with regard to *maxscore* in each item. On the run time our algorithm sees an item t in one inverted list at each scan step, and then the algorithm performs random access to pick items in other inverted lists with the same *did* as t.*did*. For each inverted list we assign a thread to implement sequential access and random access, and each thread is in charge of recording the current scan position and ready to fetch the next item. There is also another thread that is in charge of scheduling decision in interleaved manner. At the end of each scan step, this part delivers all the corresponding items of the same *did* together with their associated *Document block* to the sub-tree generator module. What's more, at each scan step, the algorithm also makes a prediction for the potential best score that the unscanned items may reach. The predication for *bestscore* is calculated as follows:

$$bestscore := \sum_{i=1\ldots m} high_i$$

Here m is the number of inverted lists and $high_i$ is the *maxscore* of each item located at the upper bound in the unvisited parts of the index lists. We will use *bestscore* together with *worstscore* generated from step 3 to implement candidate pruning and early termination.

2) The sub-tree generator module is to generate the corresponding SLCA nodes from Dewey ID lists and also calculates the corresponding score for the SLCA result nodes by summarizing all the individual score of each keyword. The sub-tree generator is implemented in IL (Index Lookup Eager Algorithm) algorithm raised in [4]. At the same time, to get the score for each SLCA result, we aggregate its partial score for each keyword by tracing the linked list for all ancestor tags corresponded to each item in *Document block*s.

3) The top-k answer selector is to preserve the k highest scores evaluated at current scan step. In our realization, we employ a min-heap for dynamic value change to generate current top-k results from last scan step's top-k results combined with the newly calculated results from sub-tree generator.

```
STA algorithm
for all index lists $L_i$ (i=1 …m) do
  item := (did, maxscore, offset)   // scan next item
  id := item.did
  for j = 1…m
    $item_j$ := $RA(list_j, id)$          // get all items with did =id
    $S_j$ := $getDocumentBlock(item_j)$
    $DList_i$ := $getDeweyList(S_i)$
  end for
  bestscore := $\sum_{j=1\ldots m} high_j$
  SLCAResult := $IL(DList_1, …, Dlist_m)$
  getScore(SLCAResult)
  TopKResult := getTopK(TopKResultOld, SLCAResult)
```

```
13    worstscore := getMin(TopKResult)
14    TopKResultOld := TopKResult
15    if worstscore <bestscore
16    Output(TopKResult)
17    else
18    return to 2
19 end for
```

We take the minimal score in the top-k answers at current scan step as *worstscore*. Now we get the *worstscore* of the current top-k results and the *bestscore* for predication of best score among unvisited items from step 1. Thus each time at the end of step 3 we would apply candidate pruning and test for possible early termination. If *worstscore* is no less than *bestscore*, the algorithm can safely terminate and output the top-k results in top-k answer selector. Or else the algorithm goes back to step 1 to continue the next scan step.

V. EXPERIENCE EVALUATION

In our experiments, we use DBLP data set for our experiments. We use java for our implementation on a 2.00 GHz server with 8.00 GB of RAM. We have evaluated and compared naive approach to get top-k results (naive approach means just going through all the inverted lists to get global scores of all items for getting the top-k results), XRank and our STA algorithm with SLCA as query semantic by a variety of keywords with different frequencies.

Fig. 4 and 5 show the response-time comparison of naive approach, XRank and STA algorithm for different groups of keyword frequencies. The response time for Fig. 4 and 5 is the average of response time after several executions. Fig. 4 and 5 are corresponding to low and high frequency keyword queries separately.

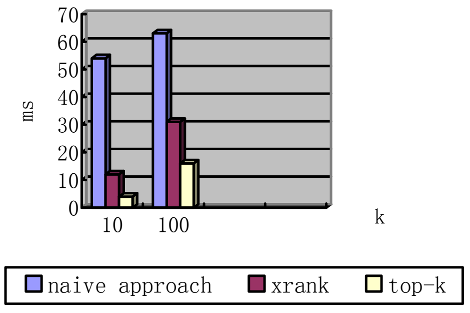

Fig. 4 Comparison with keywords of low frequency

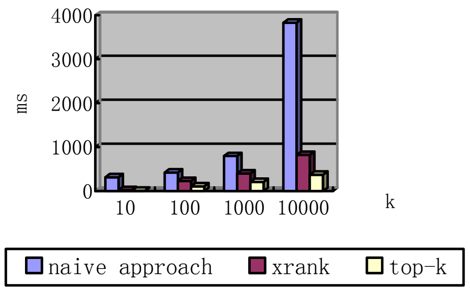

Fig. 5 Comparison with keywords of high frequency

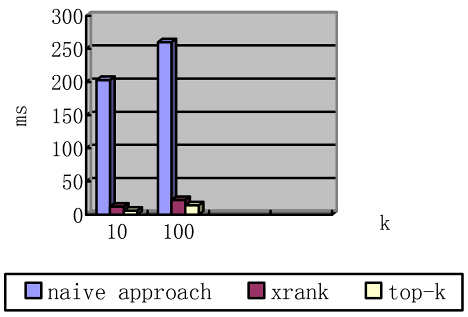

Fig. 6 Comparison with keywords of mixed frequency

In Fig. 6, each query contains two keywords; one is low frequency keyword, while the other is high frequency keyword. From Fig. 6 we could see that with this combination of keywords with mixed frequency, STA algorithm and XRank also perform better than naive approach.

VI. CONCLUSION AND FUTURE WORK

We have presented the design, implementation and evaluation of our algorithm for adaptive top-k method applied in SLCA-based XML keyword search. Our experimental evaluation shows that our index for XML data and associated algorithm offer significant performance benefits. However, there are several avenues to improve the algorithm as future work. 1) Our algorithm considers the XML data is hierarchical, taken as tree-based model. For structured (or semi-structured) data, XML data model may be a graph with the consideration of IDREFs and XLinks [3]. 2) There may be some situation that restricts random access or totally forbidden. Also, we will make a comparison about how and when to apply random access properly. 3) We will incorporate more query semantics into our algorithm [8].

VII. ACKNOWLEDGMENTS

This work was supported in part by the National High Technology Research and Development Program of China (863) under Grant Nos.2009AA01Z136 and 2009AA01Z143.

REFERENCES

[1] Martin Theobald, Ralf Schenkel, Gerhard Weikum, An efficient and versatile query engine for TopX search. In VLDB, 2005

[2] Martin Theobald, Holger Bast, Debapriyo Majumdar, Ralf Schenkel, Gerhard Weikum: TopX: Efficient and Versatile Top-k Query Processing for Semistructured Data. VLDB J. 17(1): 81-115, 2008

[3] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram. XRANK: Ranked Keyword Search over XML Documents. In SIGMOD, 2003.

[4] Y. Xu and Y. Papakonstantinou. Efficient Keyword Search for Smallest LCAs in XML Database. In SIGMOD, 2005.

[5] Z. Liu and Y. Chen. Identifying meaningful return information for xml keyword search. In SIGMOD, 2007.

[6] T. T. Chinenyanga and N. Kushmerick. Expressive retrieval from XML documents. In SIGIR 2001.

[7] N. Fuhr and K. Großjohann. XIRQL: A query language for information retrieval in XML documents. In SIGIR 2001, pages 172–180, 2001.

[8] Ziyang Liu and Yi Chen: "Reasoning and Identifying Relevant Matches for XML Keyword Search." In 34th VLDB / PVLDB Journal, Vol. 1, 2008.