XIOTR :A Terse Ranking of XIO for XML Keyword Search

Xia Li
School of Computer Science and Technology Northwestern Polytechnical University, Xi'an, China
e-mail: lixia@nwpu.edu.cn

Zhanhuai Li, Qun Chen and Ning Li
School of Computer Science and Technology Northwestern Polytechnical University, Xi'an, China
e-mail: {lizhh, chenbenben,lining}@nwpu.edu.cn

***Abstract*—The emergence of the Web has increased interests in XML data because that XML has flexible structure. Keyword search has attracted a great deal of attention for retrieving XML data because it is a user-friendly mechanism. But Keyword search is hard to directly improve search quality because lots of keyword-matched nodes may not contribute to the results. And in many applications, the goal is to find such related results that best match a set of keywords, the keywords occur location may not be consided. XML includes rich semantic information, these semantics are helpful to information retrieval process. The existing approaches of keyword search usually first generate all possible results composed of relevant tuples and then sort them based on their individual ranks. This paper investigates the compelling problem of how to take advantage of XML semantics to improve keyword search quality. We design an XML keyword search approach, that can derive the keyword query and generate a set of effective structured queries by analyzing the given keyword query and the schemas of XML data sources. Furthermore, we provide a terse algorithm to computing the rank score of the structured queries, then we can sort the results easily. We have implemented our method on real datasets and the experimental results show that our approach achieves both high recall and precise when compared with existing proposals.**

***Index Terms*—XML; Keyword Search; Structured Query; Rank**

I. INTRODUCTION

The Extensible Markup Language (XML) is becoming the dominant standard for exchanging data over the World Wide Web. As XML becomes the standard for representing web data, how to perform effective information retrieval on XML data has attracted much research interests in recent years[1-6]. The INitative for the Evaluation of XML Retrieval (INEX)[1], for example, was established in April, 2002 and has prompted XML researchers worldwide to promote the evaluation of effective XML retrieval. Keyword search is a proven and widely accepted mechanism for querying in document systems and World Wide Web. One of the key advantages of keyword search querying is its simplicity – users do not have to learn a complex query language, and can issue queries without any prior knowledge about the structure of the underlying data. Since the keyword search query interface is very flexible, queries may not always be precise and can potentially return a large number of query results, especially in large document collections. there are many unrelated results of a keyword query due to the lack of clear semantic relationships among keywords.

If users know the query languages and the structure of the XML data to be retrieved, they are able to issue a structured query, such as NEXI, XPath[2] and XQuery[3]. The desired results can be effectively retrieved because the structured query can convey complex and precise semantic meanings. Nevertheless, there are many situations where structured queries may not be applicable, such as, a user may not know the data schema, or the schema is very complex such that a query can not be easily formulated, or a user does not know how to express a search using a structured query language, as typically found in web applications.

The main shortcoming of structured queries is that user must learn the construct queries language and must know the xml data structure. Nevertheless, there are many situations where keyword search on XML documents is highly desirable.

Existing studies mainly focus on efficiency of keyword search on XML databases[7, 8], and accordingly, how to discover the structure clue from the input keywords so as to improve the precision is urgent to investigate. We emphasize the precision of keyword search on XML databases in this paper, which is at least as important as efficiency. Our objective is to provide a general method to retrieve semi-structured data efficiently.

To achieve our goal, we have published a paper [9], in the paper, we introduced a novel concept, called XIO(XML Information Object), which is a smallest

* Supported by the National Natural Science Foundation of China under Grant No. 60803043, 60970070 ; the National High-Tech Research and Development Plan of China under Grant No. 2009AA1Z134.

[1] http://www.inex.otago.ac.nz/

[2] http://www.w3.org/TR/xpath

[3] http://www.w3.org/XML/Query

doi:10.4304/jsw.6.1.156-163

meaningful XML twig as an information object result. And we introduced the algorithm to score the XIO similarity between two XIO. Then we give an efficient and adaptive keyword search strategy, called XIOF (XML Information Object Finder), to infer a set of XIO from a pure keyword query entered by a user who even has no any knowledge of the schema, then to answer pure keyword queries over semi-structured data XML by translating the XIO to effective structured queries, which can improve the performance of keyword search greatly by specifying the precise contexts of the constructed structured queries.

Since the keyword search query interface is very flexible, queries may not always be precise and can potentially return a large number of query results, especially in large document collections. there are many unrelated results of a keyword query due to the lack of clear semantic relationships among keywords. Consequently, an important requirement for XML IR is to rank the query results so that the most relevant results appear first.

The paper [9] did not conside the ranking of the queries results. In this paper, we will give a terse algorithm,called XIOTR (XML Information Object Terse Ranking)**,** to computing the XIO's rank score, and according the rank score of the structured queries to sort the queries results.

A. Our Contributions

To the best of our knowledge, the XIOTR proposed in this paper is the first approach that provides all the following features. The contributions of our work include:

(1) We introduce the notion of XIO, which is smallest meaningful information object as a twig of XML data, and propose the algorithm to score similarity of two XIO, which can dispose off the unrelated result.

(2) For different data sources, XIOF can infer different XIO from a given keyword query and the special XML data, which can be used to construct adaptive structured queries.

(3) Provide a terse algorithm, called XIOTR , to computing the rank score of XIO, so that to sort the structured queries, to rank the query results so that the most relevant results appear first.

(4)We conducted an extensive performance study using real data sets with various characteristics. an experimental evaluation of XIOTR and a comparison with alternative approaches

B. Paper Organization

The remainder of this paper is organized as follows. We discuss the related work and our motivation in Section II. In Section III, we present the data model and basic notations of XIO, then describe the XIOF algorithms. In Section IV, we present a new inverted list index structures and associated query processing algorithms for evaluating XML keyword search queries . an experimental evaluation of XIOTR and a comparison with alternative approaches are provided in Section IV. Finally we conclude the paper in Section V.

II. Related Work and Our Motivation

A. Related Work

Efficient evaluation and ranking of XML path conditions is a very fruitful research area. Solutions include structural joins[10], the multi-predicate merge join[11], the Staircase join based on index structures with pre- and post order encodings of elements within document trees[12] and Holistic Twig Joins[13]. A path stack algorithm, is probably the most efficient method for twig queries using sequential scans of index lists and linked stacks in memory. However, it does not deal with uncertain structure and does not support ranked retrieval or top-*k*-style threshold-based early termination. Vagena et al. [14] apply structural summaries to efficiently evaluate twig queries on graph-structured data, and Polyzotis et al. [15] present an efficient algorithm for computing (structurally) approximate answers for twig queries. Li et al. [16] extends XQuery to support partial knowledge of the schema. None of these papers considers result ranking and query optimization for retrieving the top-*k* results only. Information retrieval on XML data has become popular in recent years; Ref. [17]gives a comprehensive overview of the field. Some approaches extend traditional keyword style querying to XML data[18, 19]. Full-fledged XML query languages with rich IR models for ranked retrieval were introduced in Refs. [20,21]. Extensions of the vector space model for keyword search on XML documents developed in Refs. [20], whereas Li et al. [22] use language models for this purpose. Vague structural conditions were addressed in Refs. [23], Amer-Yahia et al.[24] combined this theme with full-text conditions, and Amer-Yahia et al. [25] proposed an integrated scoring model for content and vague structural conditions. More recently, various groups have started adding IR-style keyword conditions to existing XML query languages.

Keyword search over databases allows users to find pieces of information without having to write complicated structure queries. In particular, an incompletely specified query may return too many results. Nevertheless, there are many situations where structured queries may not be applicable, such as a user may not know the data schema, or the schema is very complex so that a query cannot be easily formulated or a user prefers to search relevant information from different XML documents via one query.

Given a keyword query and an XML data source, most of related work[6, 16, 26], first retrieve the relevant nodes matching with every single keyword from the data source and then compute LCA or SLCA[27] of the nodes as the results to be returned. XRANK[19] and Schema-Free XQuery[16] develop stack-based algorithms to compute LCAs as the results. [6]focus on the discussions how to infer return clauses for keyword queries XML data. [26] takes the valuable LCA as results by avoiding the false positive and false negative of LCA and SLCA. The study of query relaxation can also support structured queries when users cannot specify their queries precisely, Xbridge[28] can derive the semantics of a keyword query and generate a set of effective structured queries by

analyzing the given keyword query and the schemas of XML data sources. But in [28] the keyword must consists of the forms l : k, l : * or l : ? , where l is a label and k is a term, user have to know the labels in the xml data or know the node in the schemas of XML data sources, in other words, it is not the pure keywords query.

About the results rank, despite the success of HTML-based keyword search engines, certain limitations of the HTML data model make such systems ineffective in many domains. These limitations stem from the fact that HTML is a presentation language and hence cannot capture much semantics. The XML data model addresses this limitation by allowing for extensible element tags, which can be arbitrarily nested to capture additional semantics.

Given the nested, extensible element tags supported by XML, it is natural to exploit this information for querying. One approach is to use sophisticated query languages such as XQuery to query XML documents. While this approach can be very effective in some cases, a downside is that users have to learn a complex query language and understand the schema of underlying XML. An alternative approach, and the one we consider in this paper, is to retain the simple keyword search query interface, but exploit XML's tagged and nested structure before query processing.

XML and HTML keyword search queries differ in how query results are ranked. HTML search engines such as Google usually rank documents based (partly) on their hyperlinked structure[29]. Since XML keyword search queries can return nested elements, ranking has to be done at the granularity of XML elements, as opposed to entire XML documents. For example, different papers in the XML document in Figure 1 can have different rankings depending on the underlying hyperlinked structure. Computing rankings at the granularity of elements is complicated by the fact that the semantics of containment links (relating parent and child elements) is very different from that of hyperlinks. Consequently, techniques for computing rankings solely based on hyperlinks [29] are not directly applicable for nested XML elements.

The notion of proximity among keywords is more complex for XML. In HTML, proximity among keywords translates directly to the distance between keywords in a document. However, for XML, the distance between keywords is just one measure of proximity; the other measure of proximity is the distance between keywords and the result XML element. As an illustration, consider the keyword search query "Design Wasserman". Although the distance between the keywords "Design" (line 6) and "Wasserman" (line 10) is small, the XML element that contains both the keywords (the < article > element in line 5) is not a direct parent of either keyword, and is thus not very proximal to either keyword. Thus, for XML, we need to consider a two-dimensional proximity metric involving both the keyword distance.

```
<issues>
 <issue>
 <volume>11</volume>
 <number>1</number>
  <articles>
   <article>
    <title>Annotated Bibliography on Data Design.</title>
   <initPage>45</initPage>
   <endPage>77</endPage>
   <authors>
    <author position="00">Anthony I. Wasserman</author>
    <author position="01">Karen Botnich</author>
  </authors>
 </article>
 …
 <journal>
  <title>Database Directions III Workshop Review.</title>
   <initPage>8</initPage>
   <endPage>8</endPage>
   <authors>
    <author position="00">Tom Cook</author>
   </authors>
   </journal >
  </articles>
 </issue>
 </issues>
```

Figure 1. Example of XML document

B. Our Motivation

The above novel aspects of XML keyword search have interesting implications for the design of a search engine. In this paper, we describe the algorithm, implementation and evaluation of XIOTR built to address the above requirements for effective XML keyword search.

XIOTR aims to bridge keyword search and structured queries over XML information retrieval (IR). From a user viewpoint, it provides a keyword search interface, it allows users to search the information they are interested in without learning a complex query language or knowing the XML structure . From an IR viewpoint, XIOTR provides ranked retrieval based on a scoring function throw structured queries The key point, however, is that TopX combines these two viewpoints into a unified software system, in this paper ,with emphasis on XML ranked retrieval.

III. PRELIMINARY AND XIO SIMILARITY SCORE ALGORITHM

In this section, we describe our representation of XML data and define some basic notations to explain our proposed index.

A. Preliminary

Definition 1. (XML Tree): An XML Tree is defined as $T=(N,E,r)$ where N is a finite set of nodes, representing elements and attributes of the schema T, $N = NE \cup NV$, *NE* is set of leaf nodes, *NV* is set of middle nodes; E is set of directed edges where each edge $e(n_1,n_2)$ represents the parent child relationship between the two nodes $n_1,n_2 \in N$, denoted by $Parent(n_2)=n_1$ (node n_1 is the parent of node n_2) or $n_2 \in Child(n_1)$ (node n_2 is one of the

child node of n_1), r is the root node of the tree T. For example, in Fig.2 node 0 is root, $Parent(0.1.2) = 0.1$ and $0.1.3 \in Child(0.1)$.

In many applications, the goal is to find xml information objects related to xml dataset that best match a set of keywords. For example, as shown in Fig. 3, one might want to find the article which author is Tom, the node article and the node journal will be return as xml information object. When user input the keyword "author Tom", we can infer user want to find a XIO named article or journal rather than to find the attribute node author or the node author which is an attribute of the XIO(article). Actually, each xml document can be regarded as an xml information object, the root node is the summarization of the XML contents, and each XML can be regarded as a big XIO which contains many types of small XIO. We can see the detailed definition is shown below.

Definition 2. (XML XIO): Which is a smallest meaningful XML twig as an information object result. Given a set of labels $l_i \,|\, 1 \le i \le n$ and an XML schema tree T, a XIO is defined as the root node of the sub-tree T_{sub} of T, such that T_{sub} contains at least one schema node labeled as $l_i, ..., l_n$

Definition 3. (Alias XIO) In $T = (N, E, r)$, there are XIO_1 and XIO_2 are T_{sub} of T, the XIO_1 and XIO_2 is alias XIO, if the following two conditions hold:

1) $Root(XIO_1) \neq Root(XIO_2)$
2) $SIM(XIO_1, XIO_2) \ge \theta$

The $Root(XIO_1) \neq Root(XIO_2)$ is the root node label of XIO. $SIM(XIO_1, XIO_2)$ is the similarity score of the two XIO.

B. XIO Similarity Score Algorithm

Because the root node label of XIO is different or the expression of certain properties in different ways, we may mistakenly infer the same type XIO as different type XIO. Typical example is the existence of an alias object. For example, as shown in Fig.3, the node article and the node journal should be regarded as the same type of XIO. In order to avoid mistakenly judged the same type of XIO into different type XIO, we provide an algorithm to computing the similarity score between two XIO, as equation 1.

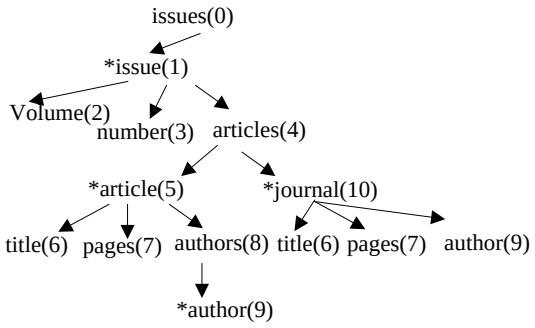


Figure 2. Example of XML document Schema

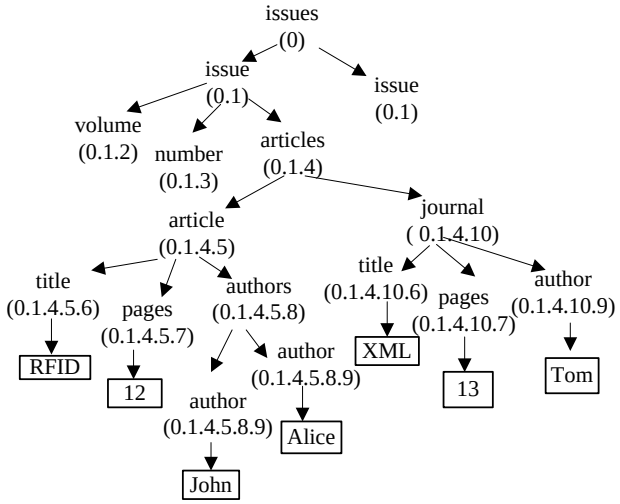


Figure 3. Example of XML document Tree

The structure of same type XIO should be similar. Therefore, we consider two factors to compute the XIO similarity, the semantic and the structural information of XIO. That is, taking into account XIO contains the node information, and to consider XIO contains hierarchy of node. The similarity score algorithm of two XIO as follows[9]:

$$SIM(XIO_1, XIO_2) = SIM\_N(XIO_1, XIO_2) * SIM\_L(XIO_1, XIO_2) \quad (1)$$

In (1), XIO_1 and XIO_2 are two twigs of T, $SIM(XIO_1, XIO_2)$)is the similarity of XIO_1 and XIO_2, $SIM\_N(XIO_1, XIO_2)$ is the node similarity of XIO_1 and XIO_2, detail in (2), and $SIM\_L(XIO_1, XIO_2)$ is the node-level similarity of XIO_1 and XIO_2, detail in (3).

$$SIM\_N(XIO_1, XIO_2) = \frac{|Root(XIO_1) \cap Root(XIO_2)|}{|Root(XIO_1) \cup Root(XIO_2)|} \quad (2)$$

In (2), $Root(XIO_1)$ is the label of the node which is contained by XIO_1. When the number of two XIO contains the same node is more, the result value is greater. If all nodes of the two XIO are same, the node similarity score is 1. Thus, $0 \le SIM\_N(XIO_1, XIO_2) \le 1$

$$SIM\_L(XIO_1, XIO_2) = \sum_{i=1, j=1}^{n} \frac{Min(XIO_{1L_{ni,nj}}, XIO_{2L_{ni,nj}})}{Max(XIO_{1L_{ni,nj}}, XIO_{2L_{ni,nj}})} \quad (3)$$

In (3), $XIO_{1L_{ni,nj}}$ is the path number between node *ni* and node *nj* in XIO1. When the count of two XIO contains the same path is more, the result value is greater. If all paths of the two XIO are same, the node-level similarity score is 1. Thus, $0 \le SIM\_L(XIO_1, XIO_2) \le 1$

C. An Adaptive Search Strategy

XIOF infer a set of XIO from a keyword query by analyzing the structure of XML, then to perform

structured queries using of existing structured query system, such as saxon[4] by translating a XIO to effective structured queries. There are three key questions need to address: (1) How to judge a XIO is meaningful. (2) How to calculate the similarity of XIO to merging similar XIO. (3) How to infer target XIO from the keywords queries and the special xml source. (4) How to score the ranking of queries results. The question(1~3) have been discussed in paper[9] , We brief describe them here, and we will emphasize on the fourth question in the follow subsection.

In XML file, there are three types data element, attribute and text, the element and attribute are structure information, and the text is content information. The element and attribute play the same role as semantic information to limit the search target, so we can regard element and attribute as the search term. The text is the search target content. Same as, the user keywords query can be divided into search term (abbreviate as s.t.) and search content (abbreviate as s.c.). If the keywords query contains s.t., we can infer the XIO structure through scan the xml schema, otherwise, if the keywords query only contains s.c. we can not infer the XIO structure directly. We describe the two kinds keywords query in this section, one is contains s.t. another is only contains s.c.

Given a list of label(as s.t.) and a XML tree T, an XIO of these labels can be represented with a sub-tree of T such that it contains the node labeled. We will derive the returned nodes only by identifying the types of the XIO. For any $XIO \in T$, if XIO is meaningful, we can determine that XIO can be taken as a return because the XIO represents an independent meaningful information object at the conceptual level. However, if XIO is un-meaningful, the XIO may not be returned. In this case, we probe its ancestor nodes until we find its nearest ancestor XIO which is meaningful. For example, as shown in Fig.3, in a digital library application, consider keyword query q2 (title xml Tom) over the XML document. We are able to infer user want to find a article or journal since the keywords query contains title (as s.c.), $title \in NT$, so we probe its ancestor nodes article or journal as the target XIO, which attribute has the node title.

However, if user issues a no structure query, such as q(xml Tom) which only contains s.c but no s.t. In this case we may not infer the XIO directly. We know, in a XML tree T, any s.c. must be contained in a text node, and any text node must be a content of an element or attribute node. So, we can have an inverted table of eigenvalue to record the relation between text and NV node based on a XML document, and we can have the structure information between NV node and NT node based on a XML schema also. So if a query no structured information, we can confirm the corresponding s.t. through scanning the inverted table of eigenvalue, then, infer a XIO based on the corresponding s.t. Such as q(xml Tom), the "xml" is contained by a text node which parent node is Title, and "Tom" is contained by a text node which parent node is author, we can trust the target XIO is article or journal which is the parent node of title and author.

[4] http://saxon.sourceforge.net/

D. Score Rank Algorithm

Almost each IR algorithm has to rank the query results so that the most relevant results appear first. According to above description, a query keywords and corresponding XML document, may infer a group of users demand information retrieval results. For sorting, first to sort the XIO, retrieval results will be in accordance with the XIO classification, a XIO retrieval results correspond to a group of retrieval results. For information retrieval results according to the classification of object, and information object, so the user has sort for easy positioning information retrieval results. The sorting algorithm as follows.

Definition 4. (Attribute Value Weight): denoted as $\omega\_t(v_i, t_i)$. Since XML keyword search queries can return nested elements, ranking has to be done at the granularity of XML elements, as opposed to entire XML documents. According the theory of TF/IDF, we get the attribute weight algorithm as following:

$$\omega\_t(v_i, t_i) = \frac{tf(v_i, t_i) \times idf(v_i, t_i)}{\max\{tf(v_i, t_i) \times idf(v_i, t_i) \mid 1 \le i \le n\}} \quad (4)$$

In (4), $tf(v_i, t_i)$ is the term frequency of node v_i which contains the keyword t_i. $idf(v_i, t_i)$ is the inverse elements frequency of the node v_i which contains the keyword t_i ,

$$idf(v_i, t_i) = \log(N / n_i + 0.01) \quad (5)$$

In (5), the N is the total number of the words which are contained in the text belong to the node v_i. When the specified word appear frequency is more in the node v_i, the result value is greater. In the node v_i , the number of different words is less, the result value is greater. If there is only one word in the text of the node v_i , the attribute value weight is 1. Thus, $0 \le \omega\_t(v_i, t_i) \le 1$

Definition 5. (Attribute Structure Weight): denoted as $\omega\_e(v_i, v_m)$. v_m is the root node of the XIO_m, obviously, the path between node v_i and the root node v_m of XIO_m is short , the weight should be larger, we get the attribute structure weight algorithm as following:

$$\omega\_e(v_i, v_m) = \frac{1}{LthOfPath(v_i, v_m) + 1} \times \frac{tf(v_i, v_m) \times idf(v_i, v_m)}{\max\{tf(v_i, v_m) \times idf(v_i, v_m) \mid 1 \le i \le n\}} \quad (5)$$

In (4), $tf(v_i, v_m)$ is the term frequency of node v_i which contains the keyword t_i. $idf(v_i, v_m)$ is the inverse elements frequency of the node v_i which contains the keyword t_i , $idf(v_i, t_i) = \log(N / n_i + 0.01)$, in here, the N is the total number of the words which are contained in the text belong to the node v_i . $LthOfPath(v_i, v_m)$ is the path between the node v_i and the XIO_m root node v_m , Obviously, $0 \le \omega\_e(v_i, v_m) \le 1$

Definition 6. (Similarity Score between XIO and Q): denoted as $Score(Q, XIO_i)$. Because the XIO the focus factor are Attribute Value and Attribute Structure , so we can get the algorithm of Similarity Score between XIO and Q as following:

$$Score(Q,O_i)=\frac{1}{n}\times[\sum_{i=1}^{k}\omega\_e(v_i,v_m)+\sum_{i=1}^{n-k}\omega\_t(v_i,t_i)] \quad (6)$$

In (6), n is the number of nodes which are contained in XIO_i, n is the number of the keywords which entered by user for quering, k is the number of search content, n-k is the number of search term Obviously, $0 \le Score(Q,O_i) \le 1$.

IV. EXPRIMENTS

We begin by describing the XML data sets and querys workloads used in the experiment. We empirically compared the performance of our *XIOTR* with the SLCA which based on Dewey ID on real-life data sets Sigmod[5], dblp[6] and auction[4]. We present the detail information of data set in Table I. The experiments were performed on 2.8GHz Pentium IV processor with 2GB of main memory and 250GB of disk space, OS is MS-Windows XP. We implemented both the SLCA based on Dewey ID and XIOTR in Java programming language. The index was stored in MySQL5.0. We present the test cases and the experimental results in Table I, There are 18 test cases from three data sets, QS1 ~QS6 is the test case of Sigmod, QD1~QD6 of dblp and QA1~QA6 of auction. In Fig.4, processing time (ms) is the average execution time for obtaining the structure language from keywords language, every process have performed 5 times.

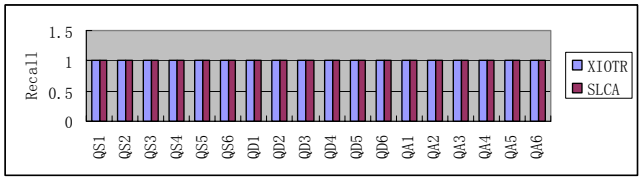
A. Performance Result

In our experiments, we found that XIOTR shows significantly better performance. Table I illustrates the experimental result of *XIOTR* . The experimental processing time compared the performance of *XIOTR* and SLCA[12] are shown in Fig.4. We observe that our algorithm achieves better search performance than the existing methods SLCA. Fig. 4 illustrates the experimental results compared the performance of XIOTR and SLCA[12]. We observe that our algorithm achieves better search performance than the existing methods SLCA. The recall rate and precision rate of the two methods are shown in the Fig. 4. In two different documents on the 16 test cases, the recall rate and precision rate are 100% on XIOF, SLCA of the recall rate was 100%, while the precision rate based on different forms in different test cases.

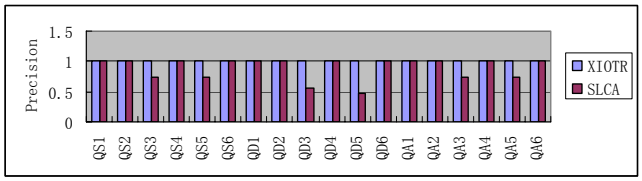
The reason of recall rate was 100% for the experiment is that the document is basically non-existent missing values, which makes the judge methods of SLCA or of XIOTR are also effective. But the precision rate, XIOTR outperform over SLCA, because XIOTR first to judge a XIO is meaningful or not before judge a XIO can be as a result, otherwise, XIOTR will dispose the meaningless result through merging similar XIO; In addition, the query be performed through Saxon which is a structure query system, and that XIOTR can provide a precise query, so our method fully reflect the high precision advantages of structured query. XIOTR method showed high precision rates, particularly in the case that the search keywords only contains s.c. The precision rate of XIOTR methods and SLCA method is differences because unmeaningful results are returned in SLCA, while XIOTR dispose the meaningless result.

TABLE I. TEST KEYWORDS QUERIES ON FILES: SIGMOD RECORD,DBLP AND AUCTION

NO.	keywork query	Create structured Query (NEXI)
QS1	author Hatfield	// ariclesTuple [about(.//author, Hatfield)]
QS2	title Description	// ariclesTuple [about(.//title, Description)]
QS3	Exploiting Gibson	//ariclesTuple[about(.//title, Exploiting Gibson)]
QS4	William	//ariclesTuple[about(.//author, William)]
QS5	Description Independence Gibson	//ariclesTuple[about(.//title, Description Independence) and about(.//author, Gibson)]
QS6	Description Languages	//ariclesTuple[about(.//title, Description Languages)]
QD1	title Normalized	//inproceedings [about(.//title,Normalized)]
QD2	booktitle SWEE Rosenthal	//www [about(.//booktitle,SWEE) and about(.//url, Rosenthal)]
QD3	author Luca ICCAD	//inproceedings [about(.//booktitle,ICCAD) and about(.//author, Luca)]
QD4	Structural Association	//inproceedings [about(.//title,Structural Association)]
QD5	booktitle Storage Retrieval	//inproceedings [about(.//title,Storage Retrieval)]
QD6	Retrieval Carole	//www [about(.//title, Retrieval) and about(.//author, Carole)]
QA1	Krishna person	//person[about (.//name, Krishna)]
QA2	Meketon	//person[about (.//street, Meketon)]
QA3	Claudine Marwedel	//person[about(.//name, Claudine) and about(.//street, Marwedel)]
QA4	person101	//persopn[about (.//id, person101)]
QA5	author person2155	//bidder[about (.//author, person2155)]
QA6	Featured	// open auction[about (.//type, Featured)]



(a) recall on XIOTR and SLCA



(b) Precision on XIOTR and SLCA

Figure 4. The Precision(a) and Recall(b)

[5] http://www.cs.washington.edu/research/xmldatasets/
[6] http://dblp.uni-trier.de/xml/

Because that XIOTR method reasonably take into account the structure of the XML in the values and relevance, and thus very well to avoid the generation of useless information. Hence, our method leads to an improvement over the existing approaches.

V. Conclusions and Future Work

XML query languages such as XQuery, XPath and NEXI use label paths to traverse the irregularly structured data, without efficient indexes, query processing can be quite inefficient due to an exhaustive traversal on XML data. But current mechanisms do not allow ordinary users to pose queries easily on semi-structured databases. We have proposed a novel index method, quick terse path index XIOTR , which contain the content and structure of the XML documents. XIOTR provides a terse index that can quickly derive the keyword query and generate effective structured queries by analyzing the given keyword query and scanning the index, hence it has a performance advantage over methods indexing either. We have conducted an extensive experimental study on real-life XML data sets. The experimental results show that XIOTR is effective, and efficient in supporting structural queries, the precision significantly outperforms the existing proposals.

Important future work includes user case studies to determine effective instantiations of our XIOTR . It is likely that a learning mechanism with user feedback will prove to be necessary to automatically dynamic index update. Another related problem of interest is partly matching for keywords queries on semi-structured databases.

VI. Acknowledgment

This work is supported by the National Natural Science Foundation of China under Grant No.60803043, 60970070; the National High-Tech Research and Development Plan of China under Grant No.2009AA1Z134 (863 Program).

References

[1] Z. Bao, T. W. Ling, B. Chen, and J. Lu, "Effective XML Keyword Search with Relevance Oriented Ranking," in Proceedings of the 2009 IEEE International Conference on Data Engineering: IEEE Computer Society, 2009, pp. 517-528.

[2] J.-M. Bremer and M. Gertz, "Integrating document and data retrieval based on XML," The VLDB Journal, vol. 15, pp. 53-83, 2006.

[3] Y. Chen, W. Wang, Z. Liu, and X. Lin, "Keyword search on structured and semi-structured data," pp. 1005 - 1010 2009.

[4] J. Feng, G. Li, J. Wang, and L. Zhou, "Finding and ranking compact connected trees for effective keyword proximity search in XML documents," Inf. Syst., vol. 35, pp. 186-203, 2010.

[5] G. Li, B. C. Ooi, J. Feng, J. Wang, and L. Zhou, "EASE: an effective 3-in-1 keyword search method for unstructured, semi-structured and structured data," in Proceedings of the 2008 ACM SIGMOD international conference on Management of data Vancouver, Canada: ACM, 2008, pp. 903-914.

[6] Z. Liu and Y. Chen, "Identifying meaningful return information for XML keyword search," in Proceedings of the 2007 ACM SIGMOD international conference on Management of data Beijing, China: ACM, 2007, pp. 329-340.

[7] F. Shao, L. Guo, C. Botev, A. Bhaskar, M. Chettiar, F. Yang, and J. Shanmugasundaram, "Efficient keyword search over virtual XML views," The VLDB Journal, vol. 18, pp. 543-570, 2009.

[8] J. F. Guoliang Li1 , Feng Lin1 and Lizhu Zhou1 "Progressive Ranking for Efficient Keyword Search over Relational Databases " pp. 193-197, 2008.

[9] X. Li, Z. Li, P. Wang, and Q. Chen, "XIOF:Finding XIO for effective keyword search in XML documents," vol. ISA 2010, pp. 99-104, 2010.

[10] S.-Y. Chien, Z. Vagena, D. Zhang, V. J. Tsotras, and C. Zaniolo, "Efficient structural joins on indexed XML documents," in Proceedings of the 28th international conference on Very Large Data Bases Hong Kong, China: VLDB Endowment, 2002.

[11] C. Zhang, J. Naughton, D. DeWitt, Q. Luo, and G. Lohman, "On supporting containment queries in relational database management systems," in Proceedings of the 2001 ACM SIGMOD international conference on Management of data Santa Barbara, California, United States: ACM, 2001.

[12] T. Grust, "Accelerating XPath location steps," in Proceedings of the 2002 ACM SIGMOD international conference on Management of data Madison, Wisconsin: ACM, 2002, pp. 109 - 120.

[13] N. Bruno, N. Koudas, and D. Srivastava, "Holistic twig joins: optimal XML pattern matching," in Proceedings of the 2002 ACM SIGMOD international conference on Management of data Madison, Wisconsin: ACM, 2002.

[14] Z. Vagena, M. M. Moro, and V. J. Tsotras, "Twig query processing over graph-structured XML data," in Proceedings of the 7th International Workshop on the Web and Databases: colocated with ACM SIGMOD/PODS 2004 Paris, France: ACM, 2004.

[15] N. Polyzotis, M. Garofalakis, and Y. Ioannidis, "Approximate XML query answers," in Proceedings of the 2004 ACM SIGMOD international conference on Management of data Paris, France: ACM, 2004.

[16] Y. Li, C. Yu, and H. V. Jagadish, "Schema-free XQuery," in Proceedings of the Thirtieth international conference on Very Large Data Bases - Volume 30 Toronto, Canada: VLDB Endowment, 2004, pp. 72-83.

[17] S. Amer-Yahia and M. Lalmas, "XML search: languages, INEX and scoring," SIGMOD Rec., vol. 35, pp. 16-23, 2006.

[18] S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv, "XSEarch: a semantic search engine for XML," in Proceedings of the 29th international conference on Very large data bases - Volume 29 Berlin, Germany: VLDB Endowment, 2003, pp. 527 - 538

[19] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram, "XRANK: ranked keyword search over XML documents," Proceedings of the 2003 ACM SIGMOD international conference on Management of data, pp. 16-27, 2003.

[20] D. Carmel, Y. S. Maarek, M. Mandelbrod, Y. Mass, and A. Soffer, "Searching XML documents via XML fragments," in Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval Toronto, Canada: ACM, 2003.

[21] T. T. Chinenyanga and N. Kushmerick, "Expressive retrieval from XML documents," in Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval New Orleans, Louisiana, United States: ACM, 2001.

[22] J. List, P. Mihajlovic, V. Mihajlovi", G. Ramirez, and A. P. Vries, "TIJAH : Embracing IR Methods in XML Databases " vol. Inf. Retr. 8(4), pp. 547 - 570, 2005.

[23] P. Schlieder and P. Meuss, "Querying and ranking XML documents " vol. JASIST 53(6), pp. 489 - 503 2002.

[24] S. Amer-Yahia, L. V. S. Lakshmanan, and S. Pandit, "FleXPath : flexible structure and full-text querying for XML " pp. 83 - 94, 2004.

[25] P. Amer-Yahia, P. Koudas, A. Marian, P. Srivastava, and P. Toman, "Structure and content scoring for XML " vol. Trondheim, Norway, pp. 361 - 372 2005.

[26] G. Li, J. Feng, J. Wang, and L. Zhou, "Effective keyword search for valuable lcas over xml documents," in Proceedings of the sixteenth ACM conference on Conference on information and knowledge management Lisbon, Portugal: ACM, 2007, pp. 31-40.

[27] Y. Xu and Y. Papakonstantinou, "Efficient keyword search for smallest LCAs in XML databases," in Proceedings of the 2005 ACM SIGMOD international conference on Management of data Baltimore, Maryland: ACM, 2005, pp. 527 - 538

[28] J. Li, C. Liu, R. Zhou, and B. Ning, "Processing XML Keyword Search by Constructing Effective Structured Queries," in Proceedings of the Joint International Conferences on Advances in Data and Web Management Suzhou, China: Springer-Verlag, 2009, pp. 88-99.

[29] J. Kleinberg, "Authoritative Sources in a Hyperlinked Environment," vol. JACM 46(5), 1999.

Xia Li (1977 -), Female, PhD, China. AnHui, Research Interests: XML IR, Data Management, Software Engineering, data management.

E-mail: lixia@nwpu.edu.cn

Tel: 13892802964

Address: School of Computer Science and Technology Northwestern Polytechnical University, Xi'an, China (710072)

Zhanhuai Li (1961 -), Male, PhD supervisor, China. ShanXi, the main academic part-time: Professional Committee of China Computer Federation Database deputy director of the China Computer Society information storage Standing Committee, Chinese Society of Astronautics Computer Professional Committee. Research Interests: Database Management System Research and Implementation; data mining; WEB data management technologies; streaming data management technology.

Qun Chen (1976 -), Male, PhD supervisor, China FuJian, Research Interests: XML IR, RFID, Data Management.

Ning Li (1978 -), Female, PhD, China. ShanXi, Research Interests: Software Engineering, Data Management, Data Mining ,