A Query-oriented XML Fragment Search Approach on A Relational Database System

Atsushi Keyaki[1], Kenji Hatano[2], Jun Miyazaki[3]

[1]Graduate School of Culture and Information Science, Doshisha University, 1-3 Tatara-Miyakodani, Kyotanabe, Kyoto 610-0394, Japan, keyaki@ilab.doshisha.ac.jp

[2]Faculty of Culture and Information Science, Doshisha University, 1-3 Tatara-Miyakodani, Kyotanabe, Kyoto 610-0394, Japan, khatano@mail.doshisha.ac.jp

[3]Graduate School of Information Science, Nara Institute of Science and Technology, 8916-5 Takayama, Ikoma, Nara 630-0192, Japan, miyazaki@is.naist.jp

Abstract: *In this paper, we propose a query-oriented fragment search approach for efficient and effective XML search engines on relational database systems. Conventional approaches for XML fragment search have only considered statistics of XML documents stored in a database system to rank a search result. As a result, almost all XML fragment search engines have attained negative results in the research field of information retrieval (IR). To cope with this problem, considering not only statistics in the database system but also ones extracted from a query helps to improve retrieval accuracy of XML fragment search engines; however, it is also faced with a problem related with their efficiencies. Therefore, we have to devise ways of achieving both highly-relevant search results and efficient query processing. We implemented our XML search engine adopting the query-oriented fragment search approach, and found that it allows users to search XML fragments more accurately and effectively than conventional ones on our XML search engine.*

Keywords: XML search engine, efficient and effective, scoring method for XML fragments, RDB

1. Introduction

Extensible Markup Language (XML) [1] is becoming widely used as a standard document format in many application domains. In the near future, we believe that a greater number of documents will be produced in XML. Therefore, in a similar way to the development of Web search engines, XML search engines will become very important tools for users wishing to explore XML documents.

It is said that the XML search engines have two outstanding features compared with traditional search engines for documents. For example, a search result of an XML search engine is usually a list of XML fragments[1] sorted in descending order of their scores, while that of traditional ones is a list of documents. The XML fragment is just a little part of XML document, so that a user can easily find an answer to his/her information need. Queries issued into XML search engines are also different between XML and document

[1] A XML document is usually likened to a tree because XML itself is mark-up language. That is to say, the XML fragment is expressed as its sub-tree based on its mark-up.

search engines. In the case of XML search engines, the queries have two conditions on both keywords and structural constraints of XML fragments which the user needs like Narrowed Extended XPath I (NEXI) [2], XQuery [3], and XQuery Full-Text queries [4], while queries are composed of only keywords in document ones. These two features are recently focused because almost all data are compiled into an XML format. These are reasons why the big demand for XML search engines is growing. As a result, several approaches have been proposed to extend the well-established scoring methods in the information retrieval research field for searching XML fragments.

Conventional approaches for searching XML fragments take into consideration both keywords and logical structure of XML documents using traditional information retrieval techniques [5]. For example, in the context of the tf-idf scoring method [6], element scoring methods pre-count term and element frequencies for each distinct tag in stored XML documents [7] [8], while path scoring methods term and path frequencies for distinct XPaths [9] [10]. These scoring methods led to improve retrieval accuracy of XML search engines; however, we found that they tended to attach large scores to small XML fragments. In short, even if using these scoring methods, a problem related to returning irrelevant search result still remains [11].

On the other hand, we also found a problem related to performance of XML search engines if we try to solve the problem related to returning irrelevant search result described above. To solve the inefficient performance problem, many researchers have recently attracted a lot of attention to methods for efficient extracting XML fragments containing all query keywords in database research area [12] [13] [14]. These approaches firstly find the lowest common ancestor (LCA) nodes among query keywords in an XML tree, and then, the XML fragments whose root nodes are the LCAs are returned as a search result. The number of such extracted XML fragments is moderate, so that we are now able to get a search result efficiently. It is true that the XML search engines can retrieve a search result efficiently; however, the retrieval accuracy of the search engines using database techniques was lower than that of using IR techniques. Consequently, the same problem described before is also occurred in this approach.

Pondering on how to cope with these problems, we have to firstly focus on statistics which have not utilized in conventional approaches yet to improve retrieval accuracy of XML search engines. The statistics utilizing conventional IR-based approaches for searching XML fragments are invariant, because they are statically extracted from all XML fragments stored in database systems. In contrast, the statistics extracted from user's information need are variable, so that we believe that using them for searching XML fragments related with user's query helps to improve the retrieval accuracy of XML search engines. Because of utilizing statistics extracted from user's query, we label our approach *query-oriented scoring method*. At the same time, considering not only highly relevant search results but also efficient query processing should be important for developing XML search engines. In our query-oriented scoring method, though the statistics that we utilize must be calculated whenever a user issues a query into an XML search engine, we would like to apply the approach by making full use of database systems storing XML documents. We believe that XML fragments would be scored effectively and efficiently using our query-oriented scoring method because RDB-based XML search engines can reflect meaning of user’s query to a search result and utilize optimization techniques developed in database research area.

The remainder of this paper is organized as follows. In Section 2, we describe how to calculate invariant and variable statistics for scoring XML fragments in XML search engines. In Section 3, we describe how to utilize database techniques to improve the performance of XML search engines, and verify them by showing experimental results in Section 4. Finally, we introduce related work of our proposal in Section 5, and conclude this paper in the last section.

2. Invariant/Variable Statistics for Scoring XML Fragments

In XML search engines, two types of queries, which are content-only (CO) and content-and-structure (CAS) queries, can usually be used. The former is composed of only keywords, though the latter is composed of both keywords and structural constraints of XML documents which the user needs. In this section, we describe two kinds of statistics for scoring XML fragments using XML search engines in detail.

2.1 Invariant Statistics for Path-based Scoring Method

In recent years, some approaches for scoring XML fragments have already been studied. One of the most famous approaches is element-based or path-based scoring method in the vector space model [15]; however we simply explain the path-based scoring method [10] here because it has been proved to perform better than the element-based scoring method [8]. The path-based scoring method is expanded the versatility of the tf-idf scoring one [6], which has been proposed to quantify the importance of terms in XML documents. The concept of the tf-idf scoring method is that a tf-idf factor of a certain term in the document becomes large if the term appears in it many times and does not appear in others at the same time. The path-based scoring method behaves the same as the tf-idf scoring method and has been used for XML search engines. The root node of each XML fragment is identified by their positions in XML documents, so that they can be classified according to their structural constraints. Assuming that the XML fragments with the same structural constraint have the same properties, we are able to quantify the importance of terms in XML fragments with the same properties as scores calculated by path-based scoring method. That is to say, the scores of a certain term in an XML fragment becomes large if the term appears in it many times and does not appear in others with the same structural constraint at the same time.

In the path-based scoring method, the scores, which are usually called tf-ipf factors, are calculated based on invariant statistics extracted from content-and-structure conditions of XML fragments. A tf-ipf factor $S_d(s)$ of an XML fragment s is composed of the following two factors, the number of occurrence of a term t in the XML fragment ($tf(s, t)$) and inverse path frequency of the XML fragment ($ipf(s, t)$), as same as the tf-idf scoring method. Describing in detail, if T is a set of keywords in a query and s is an XML fragment in a search result, $tf(s, t)$ is the number of occurrence of term $t \in T$ in s and $ipf(s, t)$ is the natural logarithm of quotient of the number of XML fragments with the same structural constraint and the number of such answer XML fragments containing term t. We assume the independence between paths in original XML documents and combine $ipf(s, t)$ of individual XPaths. For example, given the query `//article//sec[about(., t1 t2)]`, $tf(s, t_i)$ and $ipf(s, t_i)$ ($i = 1, 2$) are defined as follows:

$$tf(s,t_i) = \frac{n(s,t_i)}{N(s)}, \qquad ipf(s,t_i) = 1 + \log_{10}\frac{M(s)}{m(s,t_i)}$$

where $n(s, t_i)$ is the number of occurrence of t_i in s, $N(s)$ is the total number of term occurrence in s, $M(s)$ is the number of XML fragments which satisfy s's structural constraint (in this case, `//article//sec`) in the XML document set, and $m(s, t_i)$ is the number of such fragments containing t_i. Therefore, the tf-ipf factor $F(s)$ of XML fragment s is defined as the following equation:

$$F(s) = \sum_{t_i \in T} tf(s,t_i) \cdot ipf(s,t_i)$$

In addition, we have already found two heuristics for calculating $F(s)$ exactly. The first one is that small XML fragments are not suitable for containing a search result of XML search engines, especially in issuing query composed of only keywords. This is because the XML fragments in a search result are supposed to be semantically consolidated granules of original XML documents. In other words, such small XML fragments are not semantically consolidated granules of XML fragments which the user needs, so that they should not be included in search results. We have already pointed out this problem and have proposed a method for deleting small XML fragments from a search result using quantitative linguistics [16]. It is easy to apply this method because we just define a threshold τ related to the total number of term occurrence $N(s)$. Currently, we are using the threshold $\tau = 25$. That is to say, our XML search engine does not return the XML fragments whose total number of term occurrence is less than 25.

In contrast, the second heuristic is that $tf(s, t)$ has a negative effect for calculating $F(s)$. That is to say, the tf and ipf factors in the path-based scoring method are not well-balanced while the tf and idf factors in the tf-idf scoring method are. This fact has already been pointed out in the early INEX workshop[2], so that we believe that $ipf(s, t)$ has an insignificant effect on $F(s)$. Liu *et al.* found the same fact and proposed a method for calculating well-balanced tf factors suitable for the ipf factors using the invariant statistics of XML documents [17]. In this paper, we adapt their method to calculate $tf(s, t_i)$ using the following equation:

$$tf(s,t_i) = \frac{otf(s,t_i)}{ntf(s)}$$

where $otf(s,t_i) = 1 + \log_{10}(1 + \log_{10} n(s,t_i))$, $ntf(s) = \left(1 - \frac{M_a(s) - M(s)}{M_a(s)} \cdot c\right) \cdot (1 + \log_{10} M_a(s))$, $M_a(s)$ is an average number of terms in the XML fragments with the same structural constraint, and c is a constant parameter. The constant parameter c was usually set to 0.2. Owing to limited space, we do not describe the details of their method (see [17]).

2.2 Variable Statistics for Query-oriented Scoring Method

Using the path-based scoring method, we are able to calculate the tf-ipf factors of XML fragments related to queries before users issue them to XML search engines. Such pre-computing scores solely rely on original

[2] Initiative for the Evaluation of XML Retrieval (INEX): `http://inex.is.informatik.uni-duisburg.de/`

```
<?xml version="1.0" encoding="UTF-8"?>
<article>
 <name id="3747">Bill Gates</name>
 <body>
  <p>
   <emph3>William Henry Gates III</emph3> (born October 28, 1955), commonly
   known as <emph3>Bill Gates</emph3>, is the co-founder, chairman and chief
   software architect of Microsoft …
  </p>
  …
  <section>
   <title>Early life</title>
   <p>
    Gates was born in Seattle, Washington, to William H. Gates, Sr., a prominent
    lawyer, and Mary Maxwell Gates. Gates was …
   </p>
   <p>
    Gates, with an estimated I.Q. of 160, excelled in elementary school, particularly
    in mathematics and the sciences …
   </p>
   …
  </section>
  …
 </body>
</article>
```

Figure 1: Sample XML documents in INEX Test Collection

XML documents and do not consider query conditions on both content and structural constraint of XML fragments which a user needs. As a result, only using the path-based scoring method is unable to calculate scores of XML fragments exactly. More concretely, let us consider the XML document given in Figure 1.

This example is extracted from the INEX test collection, which is composed of document collection, topics, and relevance assessments. It has been compiled by INEX participants since 2002. The topics are following the NEXI-style query descried in [2]. If a NEXI query like //article//p[about(., "Gates")] is issued to this example XML document, the following XML fragments would be returned as a search result:

- s_1: /article[1]/body[1]/p[1]
- s_2: /article[1]/body[1]/section[1]/p[1]
- s_3: /article[1]/body[1]/section[1]/p[2]

In existing approaches, the scores of XML fragments s_1 and $s_{2,}$ s_3 are different each other because their structural constraints are different from the standpoint of the condition of the XML document. In contrast, they should be identified from the standpoint of query condition, because these XML fragments are satisfied with both keywords and structural constraints of the query. In short, we would like to give the same scores to these XML fragments satisfied with the query. Therefore, we can account for this fact by considering the condition of the issued query and defining query-oriented scores as well as path-based scores described in Section 2.1. This idea is basically similar to a scoring method which has no longer used in traditional document search techniques. However, we believe that it helps to improve the retrieval accuracies of a search result because query condition in XML fragment search is more complex than that in traditional document search.

Now we define a query-oriented score *F'(s)* of an XML fragment. Similar to the path-based score *F(s)*, *F'(s)* is also composed of two factors, the number of occurrence of a term *t* in issued query (*tf(q, t)*) and inverse the number of returned XML fragments (*iaf(t)*). We label this score the tf-iaf factor. The term *iaf(t)* is the most important factor for calculating the query-oriented score *F'(s)* and has only been explored once in isolation [8]. In the same manner as the path-based score *F(s)*, given the query //article//sec[about(., t_1 t_2)], *tf(q, t)* and *iaf(t)* are defined as follows:

$$tf(q,t_i) = w(q,t_i)$$

$$iaf(t_i) = 1 + \log_{10}\frac{V(p)}{v(p,t_i)}$$

where $tf(q, t_i)$ is the number of occurrence of t_i in the query, *V(p)* is the number of XML fragments satisfying structural constraint *p* of the query (in this case, //article//sec), and $v(p, t_i)$ is the number of XML fragments satisfying the structural constraint *p* containing term t_i. In order to calculate $iaf(t_i)$, we also assume independence between paths in the query and combine $iaf(t_i)$ of individual paths. Therefore, a query-oriented score *F'(s)* is defined as the following equation:

$$F'(s) = \sum_{t\in T} tf(q,t)\cdot iaf(t)$$

2.3 Combining Scores and Presenting Search Result

We finally define the combination of the path-based score *F(s)* and the query-oriented score *F'(s)* for the sake of scoring XML fragment *s*. Though there are many mathematical functions which can combine multiple values [18], we use the function of their products which could give creditable results in our preliminary experiments. This idea is usually used in many application domains. Note that the XML fragments which are returned from XML search engine should also contain all kinds of query keywords in themselves, so that the XML fragments containing all query keywords should be given large tf factors. Therefore, we have to take the indicator reflecting the number of query keywords in the XML fragment into the tf factor when we combine path-based and query-oriented scores as follows:

$$S(s) = \frac{q(s)}{Q}\cdot F(s)\cdot F'(s) = \frac{q(s)}{Q}\cdot\sum_{t\in T} w(q,t_i)\cdot\frac{otf(s,t_i)}{ntf(s)}\cdot\left(1+\log_{10}\frac{M(s)}{m(s,t_i)}\right)\cdot\left(1+\log_{10}\frac{V(p)}{v(p,t_i)}\right)$$

where *Q* is the number of query keywords, and *q(s)* is the number of query keywords in *s*.

3. Database Techniques for Query Processing

We have experimentally developed an XML search engine using an RDBMS based on XRel [19]. With XRel, XML documents are typically divided into five relational tables: *document*, *path*, *element*, *token*, and *attribute*. Since our XML search engine is based on an RDBMS, a NEXI query must be translated into SQL before the execution. Figure 4 shows an example of an SQL query translated from NEXI query //article[about(., greek revolution 1821)].

```
-- NEXI:  //article[about(., greek revolution 1821)]
SELECT r0.docid, r0.path, r0.nodeid, r0.l1,
           SUM(r0.score * (1 + LN(c0.count/c1.count))) * r0.inc AS rsv
FROM (
  SELECT e.docid, e.nodeid, e.path, e.l1, e.st, e.ed, SUM(t.tod) AS score,
             COUNT(t.token) AS inc
  FROM    element e, token t
  WHERE   p.pathexp LIKE '#%/article'
       AND p.pathid = e.pathid
       AND e.nodeid = t.nodeid                                              (A)
       AND e.docid = t.docid
       AND t.token IN ('revolut', 'greek')
  GROUP BY e.docid, e.nodeid, e.path, e.l1, e.st, e.ed
  ) r0, (
  SELECT CAST(COUNT(*) AS numeric) AS count
  FROM   (
   SELECT e.docid, e.nodeid, e.dewey
   FROM    element e, path p
   WHERE   p.pathexp LIKE '#%/article'                                      (B)
        AND p.pathid = e.pathid
   GROUP BY e.docid, e.nodeid, e.dewey) sc
  ) c0, (
  SELECT CAST(COUNT(*) AS numeric) AS count
  FROM   (
   SELECT e.docid, e.nodeid, e.dewey, t.token
   FROM    element e, token t
   WHERE   p.pathexp LIKE '#%/article'
        AND p.pathid = e.pathid
        AND e.docid = t.docid                                               (C)
        AND e.nodeid = t.nodeid
        AND t.token IN ('revolut', 'greek')
   GROUP BY e.docid, e.nodeid, e.dewey ) sc
  ) c1
GROUP BY r0.docid, r0.path, r0.nodeid, r0.l1, r0.inc
ORDER BY rsv DESC;
```

Figure 4: An Example of a Translated SQL Query

The term scores using a variant of path-based tf-ipf factors are pre-computed and stored in attribute *tod* of *token* table. The term $tf(q, t_i)$ is calculated by sub-query (A) in Figure 4, and, $iaf(t_i)$ is computed by sub-queries (B) and (C). On the other hand, $q(s)$ is also calculated by sub-query (A) as COUNT(t.token). Finally, the combined score $S(s)$ is obtained by calculating SUM(r0.score * (1 + LN(c0.count/c1.count))) * r0.inc in the first SELECT clause.

As seeing Figure 4, translated SQLs tend to be complicated. If the number of XML documents increases, the performance becomes degraded because such complicated SQLs must deal with a large number of data. Therefore, we need to optimize to handle such a large number of data, even using a sophisticated RDBMS.

One good solution is to utilize the *materialized view*, which keeps pre-computed views in storage to avoid re-creating the same ones for every query. The materialized view is usually supported by RDBMSs and used as an optimization technique for OLAP applications which require many aggregation queries.

For example, translated SQL queries include sub-queries which are enclosed with dotted boxes like in Figure 4. The combined score $S(s)$ is calculated by the tables which are temporarily created by these sub-queries; however, they contain joins which are potentially high cost operations. Fortunately, the structures of the sub-queries are commonly used in any queries. If the materialized view can be applied for the temporarily created tables generated by the sub-queries, instead of re-calculating sub-queries, the cost of queries drastically reduces.

The sub-queries (A), (B), and (C) in Figure 4 should be materialized. However, it has only to create two materialized views, (A) and (B), because the sub-query (C) is completely contained by view (A). The storage cost for storing materialized views affects the systems, though the size of each materialized view is not larger than that of base tables. However, the optimized queries using materialized views can improve the performance.

Another issue in the translated SQL is that it includes backward string match such as p.pathexp LIKE '#%/article' for structural constraint of a query, where the symbol '#' in SQL matches any string. Since indexes in RDBMSs are generally designed for exact and/or forward string match, such backward string match cannot utilize indexes and require scanning all data. To avoid the inefficient string match, the string data are reversed, and then, stored and indexed in the database. We call this index scheme the reverse path index. If the reverse path index is adopted, the index can be utilized even for backward string match because it is transformed to forward string match such as p.pathexp LIKE 'elcitra/%#'. This optimization could also bring performance improvement for query processing.

4. Experiments

In this section, we conducted some experiments for the sake of the effectiveness of our query-oriented scoring method and the efficiency of our RDB-based XML search engine. In order to conduct the experiments, we used the INEX 2008 test collection, which is the most famous test collection in the research field of XML search. The document collection of the INEX 2008 test collection is the Wikipedia corpus [20]. The INEX topics have 285 queries; however only 31 CO and 39 CAS topics were assessed by the INEX 2008 participants. In our experiments, our query-oriented scoring method would improve the retrieval accuracies of the XML search engine using CAS topics; so that we do not conduct the experiment for CO ones. Using this test collection, we can evaluate effectiveness and efficiency of our XML search engine.

5.1 Evaluation of Effectiveness

Figure 5 shows recall-precision curves of path-based and our query-oriented scoring methods using the focused task of the INEX 2008 relevance assessments. The reason why we use the focused task of the relevance assessments is that it is important for users to find relevant contents as much as possible, and irrelevant contents as little as possible. This assumption is the basis of methods for evaluating the effectiveness of information retrieval systems based on recall and precision in the research field of information retrieval. That is, Figure 5 indicates that our query-oriented scoring method is better than the conventional path-based scoring one from the standpoint of retrieval accuracies of XML search engines. Especially, precisions of our approach at recall levels 0.0 to 0.4 are higher than those of conventional one. This trend was also happened in other relevance assessments, which are relevance-in-context and best-in-context tasks; however they were not so clear compared with the focused task. This fact is also found by the mean average interpolated precisions (MAiP) [21]. The MAiP of query-oriented scoring method was 0.120, while that of path-based one was 0.109; consequently, the relevance has been improved by more than 10 percents. This is because our approach could provide large scores to the relevant XML fragments satisfied with user's query by combining the path-based and query-oriented scoring methods.

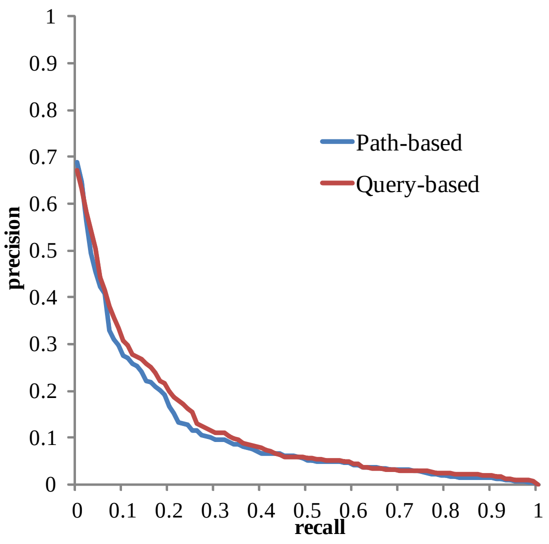

Figure 5: Recall-Precision Curves in Focused Task

Table 1: Comparisons with Other XML Search Engines in Focused Task

Participant	iP[0.00]	iP[0.01]	iP[0.05]	iP[0.10]	MAiP
Doshisha Univ.	0.6717	0.6321	0.4417	0.3068	0.1201
Renmin Univ. of China	0.5969	0.5969	0.5815	0.5439	0.2486
Queensland Univ. of Tech.	0.6232	0.6220	0.5521	0.4617	0.2134
Univ. of Amsterdam	0.6514	0.6379	0.5901	0.5280	0.2261

Table 1 compares the proposed method with three other competitive XML search engines of INEX 2008 participants. Only four systems including ours were evaluated both CAS and CO topics. iP[X] in the table indicates interpolated precision at recall level X. Our query-oriented scoring method provides higher precision values when the recall level is less or equal to 0.01. On the other hand, our proposal is inferior to others when the recall level is over 0.01. However, we assert that our proposal can be regarded as a better one in practical use, because most of users look at only a small number of top ranked results. We claim that not only the path-based scoring method but also query-oriented one can improve the ranking of the XML fragments which are maximally and exhaustively informative for users. This is because the CAS topics express users' information needs which are specified by the structural constraint of a query. The scores of the resulting fine-grained XML fragments are refined and emphasized by the query-oriented scoring method of XML fragments. As described above, we can claim that our newly-introduced scoring method can improve the effectiveness of XML search engines in perspective of relevancies.

5.2 Evaluation of Efficiency

We also evaluated the efficiency of the query processing described in Section 3 using the CAS topics in the INEX 2008 test collection. The RDBMS that we used for the evaluation was Oracle 10g R2 running on a Sun Fire X4600M2 machine which has 8 3.0GHz dual core Opteron processors, 256GB of memory, and 12TB of storage.

We first conducted the preliminary experiments on reverse path index. We found that only a few percents of performance improvement could be obtained, because other processing cost was dominant. Therefore, use of the reverse path index is regarded as an optional optimization, though it basically depends on the number of the paths included in documents. However, the reverse path index never deteriorates the performance.

On the other hand, when materialized views were used, the total execution time was 5,642 sec. However, in case of no materialized view, the total execution time became 9,210 sec. That is to say, the optimization using materialized views that we mentioned provided 1.63x faster query processing. In other words, our approach can fully utilize sophisticated relational database technologies.

5. Related Work

The application of information retrieval techniques in searching XML fragments has become an area of research in recent years. As we described before, the INEX participants have proposed many types of scoring methods for XML search engines [5]. Over the years, it has become clear that refining the level of granularity at which document structure is taken into account in pre-computing individual term scores either in the vector space model or the probabilistic model has improved retrieval accuracies. However, the query-oriented scoring method has not been explored to the extent at which we are proposing in this paper.

Fuhr *et al.* proposed a method for propagating scores of XML fragments leaf-to-root along the XML document tree [22]. However, although XIRQL, their proposed language, enables queries with a mix of conditions on both keywords and logical structure of XML documents, only the keywords are scored using path-based scoring method. Other scoring methods also use conditions on document structure to apply length normalization between structural condition of query and XML documents, to compute term scores based on element tags or structural constraints [7] [8] [9] [10]. It was reported that these methods were useful for searching XML fragments [23]; however, such methods did not use statistics of original XML documents and structural conditions of queries. At the same time, some researchers have proposed to find minimal sub-trees in XML documents containing all query keywords efficiently in database research field [12] [13] [14]. These approaches assume that the XML fragments whose root nodes are the LCAs should be answer XML fragments, so that their research purposes focused only on finding the LCA nodes in XML documents efficiently. However, their assumptions may or may not be correct, so that we have to utilize all XML fragments and to calculate their scores to develop effective XML search engines.

In this paper, therefore, we propose query-oriented scoring method based on variable statistics of stored

XML documents, and combine two scoring methods to improve retrieval accuracies of XML search engines. Moreover, adopting our simple device related to query processing of our XML search engine could handle the performance problem caused by our new ranking algorithm. We could verify the effectiveness and efficiency of above our proposals through the experimental evaluations in Section 4.

6. Conclusion and Feature Work

In this paper, we proposed a query-oriented scoring method for developing XML fragment search engines. Our main contribution is to improve retrieval accuracy and performance of XML search engines, and we confirmed that effectiveness and efficiency of our XML search engine is better than that of competitive ones. In particular, the query-oriented scoring method could make a significant contribution to improve retrieval accuracy of XML search engines. Applying database technique was also useful for efficient query processing, so that it could be said that it brought a great advantage in practical use.

In the future, we would like to address the following issues:

- As we described Section 4, our scoring method cannot be applied to CO topics, because query-oriented scoring method is calculated when a query is issued. Therefore, we have to study the meanings of *informative XML fragments* issuing CO topics to XML search engines, and have to propose an approach to improve their retrieval accuracies. We are currently formalizing to define fine-grained XML fragments for CO topics, and are going to propose a novel information retrieval model for XML fragment retrieval.
- Our new scoring method is suitable for searching relevant XML fragments to user's query; however, it takes a long time to return a search result compared with a conventional Web search engines when a user issues a query with complex structural constraint. Therefore, we have to develop an efficient XML query processing mechanism in our XML search engine.
- In order to develop an XML search engine, it is also important for users to propose not only scoring method but also a user interface for issuing XML queries and browsing search results. This is because the user interface helps users to grasp and understand search results efficiently. We are currently developing such a user interface on top of our XML search engine.

Acknowledgements

This work was supported in part by MEXT (Grant-in-Aid for Scientific Research on Priority Areas #20700227 and #21013035).

Reference

[1] Bray, T., Paoli, J., Sperberg-McQueen, M., Maler, E., and Yergeau, F. (2008). Extensible Markup Language (XML) 1.0 (Fifth Edition), http://www.w3.org/TR/xml/.

[2] Trotman, A. and Sigurbjörnsson, B. (2005) Narrowed Extended XPath I (NEXI). *Advances in XML*

Information Retrieval, pages 16-40. Springer.

[3] Boag, S., Chamberlin, D., Fernández, M. F., Florescu, D., Robie, J., and Siméon, J. (2007). XQuery 1.0: An XML Query Language, http://www.w3.org/TR/xquery/.

[4] Amer-Yahia, S., Botev, C., Dorre, J., and Shanmugasundaram, J. (2006). XQuery Full-Text Extensions Explained. *IBM Systems Journal* 45(2), pages 335-352.

[5] Amer-Yahia, S. and Lalmas, M. (2006) XML Search: Languages, INEX and Scoring. *SIGMOD Record* 35(4), pages 16-23. ACM.

[6] Salton, G. and Buckley, C. (1988) Term-Weighting Approaches in Automatic Text Retrieval. *Information Processing & Management* 24(5), pages 513-523.

[7] Cohen, S., Mamou, J., Kanza, Y., and Sagiv, Y. (2003) XSEarch: A Semantic Search Engine for XML. *Proceedings of 29th International Conference on Very Large Data Bases*, pages 45-56.

[8] Amer-Yahia, S., Koudas, N., Marian, A., Srivastava, D., and Toman, D. (2005) Structure and Content Scoring for XML. *Proceedings of the 31st International Conference on Very Large Data Bases*, pages 361-372.

[9] Carmel, D., Maarek, Y.S., Mandelbrod, M., Mass, Y., and Soffer, A. (2003) Searching XML Documents via XML Fragments. *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 151-158.

[10] Grabs, T. and Schek, H. J. (2003) PowerDB-XML: A Platform for Data-Centric and Document-Centric XML Processing. *Proceedings of the First International XML Database Symposium*, pages 100-117.

[11] Fujimoto, K., Shimizu, T., Terada, N., Hatano, K., Suzuki, Y., Amagasa, T., Kinutani, H., and Yoshikawa, M. (2006) An Implementation of High-Speed and High-Precision XML Information Retrieval System on Relational Databases. *Advances in XML Information Retrieval and Evaluation*, pages 254-267. Springer.

[12] Liu, Z. and Chen, Y. (2007) Identifying Meaningful Return Information for XML Keyword Search. *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, pages 329-340.

[13] Li, Y., Yu, C., and Jagadish, H.V. (2004) Schema-Free XQuery. *Proceedings of the 30th International Conference on Very Large Data Bases*, pages 72–83.

[14] Xu, Y. and Papakonstantinou, Y. (2005) Efficient Keyword Search for Smallest LCAs in XML Databases. *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, pages 527-538.

[15] Salton, G., Wong, A., and Yang, C. S. (1975) A Vector Space Model for Automatic Indexing. *Communication of the ACM* 18(11), pages 613–620. ACM.

[16] Hatano, K., Kinutani, H., Amagasa, T., Mori, Y., Yoshikawa, M., and Uemura, S. (2005) Analyzing the Properties of XML Fragments Decomposed from the INEX Document Collection. *Advances in XML Information Retrieval*, pages 168–182. Springer.

[17] Liu, F., Yu, C.T., Meng, W., and Chowdhury, A. (2006) Effective Keyword Search in Relational Databases. *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*,

pages 563–574.

[18] Montague, M. and Aslam, J. A. (2002) Condorcet Fusion for Improved Retrieval. *Proceedings of the 11th International Conference on Information and Knowledge Management*, pages 538-548.

[19] Yoshikawa, M., Amagasa, T., Shimura, T., and Uemura, S. (2001) XRel: A Path-based Approach to Storage and Retrieval of XML Documents using Relational Databases. *ACM Transactions on Internet Technology* 1(1), pages 110-141. ACM.

[20] Denoyer, L. and Gallinari, P. (2006) The Wikipedia XML Corpus. *SIGIR Forum* 40(1), pages 64–69. ACM.

[21] Kamps, J., Pehcevski, J., Kazai, G., Lalmas, M., and Robertson, S. (2008) INEX 2007 Evaluation Measures. *Focused Access to XML Documents*, pages 24–33. Springer.

[22] Fuhr, N. and Großjohann, K. (2004) XIRQL: An XML Query Language based on Information Retrieval Concepts. *ACM Transactions on Information Systems* 22(2), pages 313-356. ACM.

[23] Kazai, G. and Lalmas, M. (2006) INEX 2005 Evaluation Metrics. *Advances in XML Information Retrieval and Evaluation*, pages 16-29. Springer.