Ranking and Presenting Search Results in an RDB-based XML Search Engine

Kenji Hatano[1], Toshiyuki Shimizu[2], Jun Miyazaki[3], Yu Suzuki[4], Hiroko Kinutani[5], and Masatoshi Yoshikawa[2]

[1] Faculty of Culture and Information Science, Doshisha University
khatano@mail.doshisha.ac.jp
[2] Graduate School of Informatics, Kyoto University
shimizu@soc.i.kyoto-u.ac.jp, yoshikawa@i.kyoto-u.ac.jp
[3] Graduate School of Information Science,
Nara Institute of Science and Technology
miyazaki@is.naist.jp
[4] College of Information Science and Technology, Ritsumeikan University
suzuki@ics.ritsumei.ac.jp
[5] Institute of Industrial Science, The University of Tokyo
kinutani@tkl.iis.u-tokyo.ac.jp

Abstract. Conventional ranking methods for document search have considered content of documents to rank a search result. They have attained some positive results in the research area of document search; however, it has been said that content of not only documents but also queries should be utilized if users want to get a search result accurately. This fact applies to XML search engines. In this paper, therefore, we propose a ranking method for XML search considering content-and-structure conditions of both XML documents and queries. We also propose a method for presenting a search result for XML search, because it is very important for users to grasp and understand the entire search result, too. We implemented our ranking method on top of XRel, a relational database system for XML documents, and found that our proposal allows users to search XML fragments more accurately than previously proposed approaches for XML search.

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.

In the meantime, a search result of current Web search engines is usually a list of Web documents. That is, Web documents are sorted in descending order of their scores. The scores are calculated by content of Web documents, which are quantified based on the number of occurrences of terms extracted from Web

documents like the *tf-idf* scoring [2]. In the case of XML search engines, it is said that XML queries combine conditions on both content and logical structure such as Narrowed-Extended XPath I (NEXI) [3] and XQuery Full-Text queries [4]. When such queries are issued to an XML search engine, the search result is usually a list of XML fragments[6] as opposed to that of entire documents in current Web search engines. As a result, several approaches have been proposed to extend the well-established content-based scoring in some retrieval with the ability to rank XML fragments.

Conventional approaches for XML search take into consideration both content and logical structure of XML documents in order to rank XML fragments which satisfy query conditions [5]. For example, in the context of *tf-idf* scoring, element scoring precomputes *tf* and *idf* factors for each distinct tag in input XML documents [6, 7], while path scoring precomputes them for distinct paths [8, 9]. These refined scoring approaches led to improvements in the retrieval accuracy of search results consist of scored XML fragments. However, these approaches tended to attach great importance to small XML fragments, so that they caused a problem returning small XML fragments partially satisfied with users' information need in some cases [10–12]. On the other hand, it is also said that it is important for XML search engines to handle overlapping parts of XML fragments. It means that when a user grasps and understand the content of an XML fragment, the user browses ancestor of the XML fragment unconsciously. In short, users can grasp and understand the content of search results if XML search engines can indicate a list of large size of XML fragments containing small ones. Considering this fact, Clarke proposed to control overlapping by re-ranking the descendant and ancestor of search results [13]. In Clarke's approach, however, users have to browse the overlapping parts of XML fragments more than once, so that it increases the burden on users.

In order to overcome two problems described above, we propose ranking and presenting methods of XML fragments as search results for XML search engines. In our ranking method, we advocate the use of two scoring algorithms for content-only (CO) and content-and-structure (CAS) queries. The former is based on the content condition of XML documents like conventional element or path scoring methods, and is utilize statistics extracted from XML documents effectively, though the latter is based on the content-and-structure conditions of both XML documents and queries. This is because the basic idea of our ranking method has been shown to improve retrieval accuracies of search results in the research area of traditional document search. At the same time, we also insist that we devise ways of effectively presenting search results to handle overlapping parts of answer XML fragments, because XML search engines just have to decide and present one XML fragment in one way or another if there is an ancestor-descendant relationship among XML fragments in a search result. In order to verify the effectiveness of our proposals, we implemented two scoring methods on a relational database system for XML documents based on XRel [14]. Our

[6] XML fragments are easily extracted from XML documents based on their markup. That is, they are subtrees in the XML trees.

experiments on the INEX test collection show that using content-and-structure conditions of both documents and queries improves the retrieval accuracies of XML search engines.

The remainder of this paper is organized as follows. In Section 2, we describe how to calculate the scores of XML fragments based on content-and-structure conditions of both documents and queries and statistics of XML documents. In Section 3, we also describe how to present XML fragments with ancestor-descendant relationships. We report our experimental results to verify our proposal in Section 4 and related work in Section 5. Finally, we conclude this paper in the last section.

2 Our Ranking Method based on Content-and-Structure Conditions

In Section 2.1 and 2.2, we describe our two types of scoring algorithms in detail. One is for CO queries and takes the content-and-structure condition of XML documents into consideration. The other is for CAS queries and considers that of both XML documents and queries. We also explain a method for integrating two scoring algorithms in Section 2.3.

2.1 Content-and-Structure Conditions of XML Documents

As we described in Section 1, conventional methods for calculating scores of XML fragments have already studied in recent years. One of the most famous methods for calculating scores of XML fragments is element-based or path-based scoring in the vector space model [15]; however we simply explain the path-based scoring here because it has proved to perform better than element scoring [7].

The path-based scorings like the *tf-ipf* scoring [9] are expanded the versatility of the *tf-idf* scoring [2],which has been proposed to quantify the importance of terms in documents. The concept of the *tf-idf* scoring is that a tf-idf score 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 *tf-ipf* scoring behaves the same as the *tf-idf* scoring and has been used for XML search. XML fragments extracted from an original XML document are identified their XPath expressions [16], so that they are classified according to the abbreviated syntax of their XPath expressions. Assuming that the XML fragments with the same abbreviated XPath expression have the same properties, we can quantify the importance of terms in XML fragments with same properties as tf-ipf scores. That is to say, a tf-ipf score 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 abbreviated XPath expression at the same time.

In our scoring algorithm, we define a *tf-ipf* score calculated from content-and-structure conditions of XML documents. The score S_d is composed of two factors, "Term Frequency of XML fragment (tf_d)" and "Inverse Path Frequency of XML fragment (ipf_d)" as same as the *tf-idf* scoring. These factors are inspired

from one of the path-based scorings proposed in [9]. In short, if T is the set of query terms and s is an answer XML fragment in a search result, $tf_d(s\ t)$ is the number of occurrences of term $t \in T$ in s and $ipf_d(s\ t)$ is the natural logarithm of quotient of the number of XML fragments which have the same structure as s and the number of such answer XML fragments containing term t. We assume the independence between paths in original XML documents and combine $ipf_d(s\ t)$ of individual paths. For example, given the query `//article//sec[about(.,` $t_1\ t_2$`)]`, $tf_d(s\ t_i)$ and $ipf_d(s\ t_i)(i = 1\ 2)$ are defined as follows:

$$tf_d(s\ t_i) = \frac{n(s\ t_i)}{l(s)} \qquad ipf_d(s\ t_i) = 1 + \log \frac{M(s)}{m(s\ t_i)} \tag{1}$$

where $n(s\ t_i)$ is the number of occurrences of t_i in s, $l(s)$ is the length of s (total number of terms in s), $M(s)$ is the number of XML fragments in the original XML documents which satisfy s's structure, and $m(s\ t_i)$ is the number of such fragments containing t_i. Therefore, a score considering content-and-structure condition of XML documents S_d is defined as the following equation:

$$S_d(s) = \sum_{t \in T} tf_d(s\ t)\ \ ipf_d(s\ t) \tag{2}$$

In addition, we have already found two heuristics for calculating $S_d(s)$ exactly. The first heuristic is that small XML fragments are not suitable for search results in XML search engines, especially keyword search. 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, so that they should not be included in search results. We have already pointed this problem in [12], and proposed a method for deleting small XML fragments from search results using quantitative linguistics [11]. Applying this approach proposed in [11] to our XML search engine easily, we defined a threshold called the ratio of period to delate such small XML fragments from search results in [10]. The ratio of period is defined as follows:

$$r(s_e) = \frac{n_p(s_e)}{N_p(s_e)} \tag{3}$$

where $N_p(s_e)$ denotes the number of XML fragments whose tag names is s_e[7], and $n_p(s_e)$ is the number of XML fragments that end with the symbols like `.`, `?`, or `!` if the node with tag name s_e is a leaf node, or the number of XML fragments that have more than one document-centric leaf node if the node with tag name s_e is an internal node.

In contrast, the second heuristic is that $tf_d(s\ t_i)$ has a negative effect for calculating $S_d(s)$. That is to say, the tf and idf factors in the *tf-idf* scoring are well-balanced; however, the tf and ipf factors in the *tf-ipf* scoring are not well-balanced. For example, $tf_d(s\ t_i)$ is more influence over $S_d(s)$ than $ipf_d(s\ t_i)$ in experiments of the INEX test collections (from 2002 to 2005), so that we believe

[7] s_e is the tag name of an XML fragment s.

that $ipf_d(s\ t_i)$ has an insignificant effect on $S_d(s)$. Liu et al. also found the same fact and proposed an well-balanced *tf* factors suitable for $ipf_d(s\ t_i)$ based on statistics extracted from original XML documents, which were calculated by using the average number of terms of the XML fragments with the same abbreviated XPath expression $l_{ave}(s)$ and a constant parameter c as follows:

$$tf_d(s\ t) = \frac{o\_tf(s\ t)}{n\_tf(s)} \tag{4}$$

$$o\_tf(s\ t) = 1 + \log\left(1 + \log\left(n(s\ t)\right)\right) \tag{5}$$

$$n\_tf(s) = \quad 1 \quad \frac{l_{ave}(s) \quad l(s)}{l_{ave}(s)} \quad c \quad \left(1 + \log(l_{ave}(s))\right) \tag{6}$$

In this paper, we adapted their methods to original *tf-ipf* scoring and calculated $tf_d(s\ t)$ defined in equation (4). We call this scoring method "*ntf-ipf* scoring", which is extended by using statistics of original XML documents. The constant parameter c in equation (6) was usually set to 0 2. Owing to limited space, we do not describe the details of their method (see [17]).

2.2 Content-and-Structure Conditions of Queries

Using the path-based scorings, we can calculate scores of XML fragments related to queries before users issues them to XML search engines. Such precomputing scores solely rely on original XML documents and do not consider query conditions on both content and structure. As a result, only using the path-based scorings is unable to function to calculate scores of answer XML fragments exactly.

More concretely, let us consider the XML document given in Fig. 1. This example is extracted from the INEX 2007 document collection. If a NEXI query like `//article//p[about(.,"Gates")]` is issued to this example, XML fragment s_1: `/article[1]/body[1]/p[1]`, s_2: `/article[1]/body[1]/section[1]/p[1]`, and s_3: `/article[1]/body[1]/section[1]/p[2]` would return as a search result. In existing approaches, the scores of XML fragments s_1 and $s_{2\ 3}$ are different each other because their abbreviated XPath expressions are different from the standpoint of both content and logical structure of original XML documents. From the standpoint of query condition, however, they should be identified because these XML fragments are satisfied with both content and logical structure of the query. In short, we would like to give the same scores to the XML fragments satisfied with all condition of the query. Therefore, we can account for this by considering condition on content and structure in the input queries and defining scores as a function of those conditions as well as precomputed document-based scores described in Section 2.1. This idea is basically the same in traditional document search [2], and we believe that it helps to improve the retrieval accuracies of search results.

Now we define a query-based score S_q. Similarly to the document-based score $S_d(s)$, S_q is composed of two factors, "Term Frequency of Query (tf_q)" and "Inverse Answer Document Frequency of Query (iaf_q)", so that we call this scoring

```
<?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 Corporation,
      the largest software company in the world. According to ...
    </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 born
        with a million dollar trust fund set up by his grandfather, ...
      </p>
      <p>
        Gates, with an estimated I.Q. of 160, excelled in elementary
        school, particulary in mathematics and the sciences ...
      </p>
      ...
    </section>
    ...
  </body>
</article>
```

Fig. 1. A Sample XML Document in the INEX 2007 Document Collection

the *tf-iaf* scoring. iaf_q is important for calculating the query-based score S_q and has only been explored once in isolation [7]. However, the cost of calculating iaf_q can be quite expensive. Therefore, we only focus on the effectiveness of XML search engines in this paper. In the same manner as a document-based score $S_d(s)$, given the query `//article`
`//sec[about(., t1 t2)]`, $tf_q(t_i)$ and $iaf_q(t_i)$ are defined as follows:

$$tf_q(t_i) = w(t_i) \qquad iaf_q(t_i) = 1 + \log \frac{V(p)}{v(p\ t_i)} \tag{7}$$

where $w(t_i)$ is the number of occurrences of t_i in the query, $V(p)$ is the number of XML fragments satisfying the query path p (in this case, `//article//sec`), and $v(p\ t_i)$ is the number of XML fragments satisfying the query path p containing term t_i. In order to calculate $iaf_q(t_i)$, we also assume independence between paths in the query and combine $iaf_q(t_i)$ of individual paths. Therefore, a query-

based score S_q is defined as the following equation:

$$S_q = \sum_{t \in T} tf_q(t) \;\; iaf_q(t) \tag{8}$$

2.3 Our Ranking Method

We finally define the combination of a document-based score $S_d(s)$ and a query-based score S_q. This idea is inspired from the SMART retrieval system[8], which has been considered the term weights of both documents and queries. In order to combine them, the SMART retrieval system calculates their product in the same spirit as document scores described in [2].

In our method, we apply the same idea to our XML search engine. Scores of an XML fragment s related to a query is thus defined as follows:

$$S(s) = \sum_{t \in T} S_d(s) \;\; S_q = \sum_{t \in T} tf_d(s\ t) \;\; ipf_d(s\ t) \;\; tf_q(t) \;\; iaf_q(t) \tag{9}$$

3 Search Result Presentation

As we described in Section 1, it is also important for improving the retrieval accuracy of XML search engines to propose a method for presenting search results. This is because XML search should consider the overlapping parts of answer XML fragments unlike document search. Considering this fact, Clarke has proposed to control overlapping by re-ranking the descendant and ancestor of search results [13]. Compared with his approach, we propose a concept of search result presentation which is a unit of answer XML fragments and use it in our XML search engine. We believe that our search result presentation helps for users to grasp and understand the entire search results effectively compared with conventional approaches.

3.1 Search Result Presentation for XML Search

XML search engines extract XML fragments satisfied with a query from original XML documents. In other words, it remains possible that a large number of answer XML fragments are returned from XML search engines. Such answer XML fragments may be extracted from one XML fragment. For example, XML documents in the 2005 INEX document collection are scholarly articles, so that sections, subsections, paragraphs and so on are retrieved by XML search engines. Such retrieved XML fragments may overlap due to nesting structure of XML documents. This fact causes the problem to be difficult to grasp and understand the entire search results effectively.

Because of the above situation, the INEX project has demanded some kinds of search result presentations such as not *Thorough* strategy but *Focused*, *Relevant-In-Context* and *Best-In-Context* ones. While the XML search engines with the

[8] `ftp://ftp.cs.cornell.edu/pub/smart/`.

Thorough strategy can retrieve overlapping XML fragments, ones with other strategies can retrieve non-overlapping document parts containing answer XML fragments or a single document part per an XML document. That is, they firstly extract XML fragments related with queries, and then decide answer parts from extracted ones. We think that, however, these strategies also contain the problem because the XML search engines with these strategies find non-overlapping document parts using scored XML fragments in an XML document regardless of their scores. The best way to attain the most effective XML search is to extract some XML fragments related with the queries from one XML document, to generate document parts based on a unit appropriate for users, and to rank them for presenting search results. Considering these demands, we believe that an XML search engine would be more useful if it has a user interface which can handle a basic unit for XML search and can provide answers constructed from the unit. This is because it is natural for users to show answers mapped on original XML documents, and the users avoid the need to see the document parts not related with queries. In short, our XML search engine provides thumbnail of original XML documents and indicates the answer parts of XML documents directly in its user interface; in consequence, users can intuitively grasp and understand the search results[9].

In order to implement such user interface of our XML search engine, we propose a new concept called "Aggregation Granularity (AG)", which is a unit of search results determined from original XML documents. In next section, we describe our new concept in detail.

3.2 Aggregation Granularity

In conventional XML search engines, answer XML fragments are showed in their user interfaces individually on the *Thorough* strategy, so that users tend to get messed up the relationship among the answer XML fragments. In the case of our XML search engine realizing the new concept, answer XML fragments are allocated on original XML documents in its user interface. Therefore, the problem described in previous section is not caused in our XML search engine.

In some case, however, we would be better off aggregating several answer XML fragments with large scores into one document part to show the search results to users, because it is easy for users to understand the content of a search result from the viewpoint for grasping the outline of original XML document even if the score of the document part, which is also XML fragment containing the answer XML fragments, is not large. For example, a query is issued by a user, conventional XML search engines return answer XML fragments whose root nodes are gray-color elements in Fig. 2. As a result, a large number of answer XML fragments are returned, so that users cannot grasp and understand the search result. In our XML search engine, however, answer XML fragments with a certain degree of scores whose root nodes are gray-color elements in

[9] The difference between the *Focused* strategy and our proposal is to be able to highlight answer XML fragments with large scores.

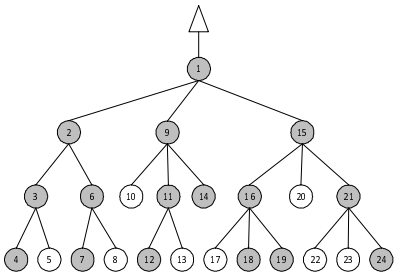

Fig. 2. Answer XML Fragments in Existing XML-IR System

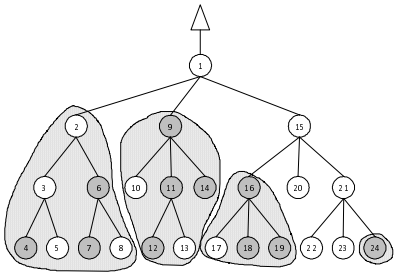

Fig. 3. Answer XML Fragments in Our XML-IR System

Fig. 3 are extracted from the search result, and then, some document parts enclosed by trajectory in Fig. 3 are constructed from them as basic units for XML search, AGs. As a result, users can grasp and understand the entire search results effectively compared with conventional XML search engines.

In this approach, the following two things become big problems. One is how to decide AGs, and the other is how to calculate the score of the document parts based on AG. To cope with the first problem, the AG can be defined if a certain standard like the threshold size, the location in original XML documents of answer XML fragments, and so on. In [18], for example, XML documents can be divided into multiple parts like physical pages, so that the AG is defined as individual pages of XML documents. Generally, XML fragments suitable for an AG tend to be located at the higher level of original XML documents, and their sizes tend to be relatively large. In short, it seems more likely that element `2`, `9`, and `15` in Fig. 3 would be first candidate of AG, and element `3`, `6`, `11`, `16`, and `21` would be second candidate. Alternatively, calculating scores of aggregated XML fragments varies in methodology. The easiest way to calculate their scores is the sum of the scores of answer XML fragments which constitute the document parts defined from AG. However, two problems above have a lot of things to be considered, so that now we are formulating the definition and score-calculation of AG. We would like to try every way possible to formulate and implement them in our XML search engine in the near future.

4 Experimental Evaluations

In this section, we conduct some experiments for the sake of the effectiveness of our proposals in our XML search engine. At the present time, an evaluation tool is not available, so that we show the experimental results using the 2005 INEX test collection[10]. This collection is composed of a document set marked up in XML, its relevance assessment, and evaluation measures. The document set contains 16,819 articles of the IEEE Computer Society's magazines and transactions published from 1995 to 2004. The size of the document set is 735MB,

[10] We could not take part in INEX 2006.

an article contains 1,532 XML nodes on average, and the average depth of a node is 6.9. The relevance assessment has two graded dimensions to express relevance of XML fragments to XML queries, "exhaustivity" and "specificity". The concept of specificity is peculiar to XML search, because it provides a measure of the size of an XML fragment as it measures the ratio of relevant to non-relevant content within the XML fragment. In order to identify relevant XML fragments to XML queries, INEX project provides two evaluation measures, recall-precision and eXtended Cumulated Gain (XCG) [19]. The recall-precision is used for evaluating the effectiveness of conventional information retrieval systems. The recall-precision in the INEX project maps the values of exhaustivity and specificity to a single scale using quantization functions [20]. On the other hand, the XCG was additionally proposed for evaluating effectiveness of XML search engines [21] because the recall-precision evaluation measure could not consider overlapping XML fragments. This problem is amply explained in [22] and can be summarized as the issue of avoiding to return both elements and their sub-elements as query results and recalculating scores on the fly when that happens. In that sense, the XCG measure accounts for both retrieval accuracy and users experience.

4.1 Evaluation of Scoring based on Document Conditions

In this evaluation, we used the recall-precision and the XCG measures to evaluate our scoring algorithm for CO queries.

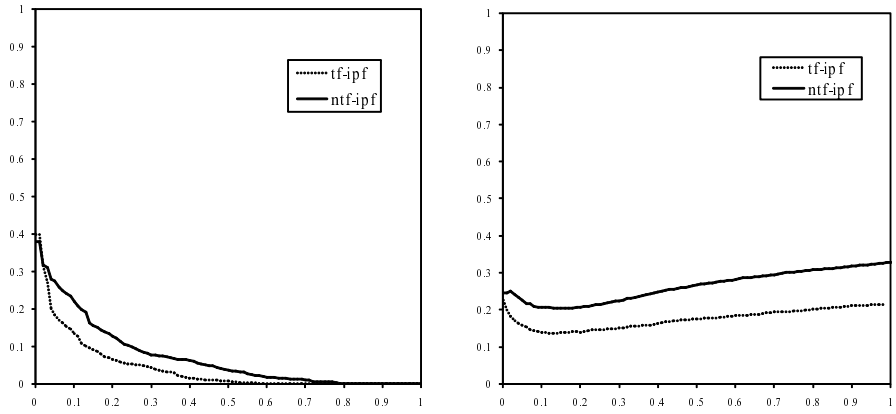

Fig. 4. Retrieval Accuracies based on Recall-Precision/nXCG

Fig. 4 shows the retrieval accuracies based on the recall-precision and the nXCG in the INEX evaluation measures. "tf-ipf" in Fig. 4 is the original *tf-ipf* scoring, "ntf-ipf" is the scoring method defined in equation (2) where $tf_d(s\ t)$ is

redefined in equation (4)[11]. Fig. 4 speaks that the *ntf-ipf* scoring could retrieve more relevant XML fragments than the *tf-ipf* one in the *Thorough* strategy. In short, we can verify the effectiveness of the *ntf-ipf* scoring for the CO queries. We also noticed that we have to formulate not only the ipf factor but also the tf factor for effective XML search, because the original *tf-ipf* scoring has never configured the tf factor in the *tf-idf* scoring for document search. As a result, it is important for effective XML search to use the statistics extracted from original XML documents and to formulate the scoring algorithm.

4.2 Evaluation of Scoring based on Query Conditions

In this evaluation, we also used two evaluation measures to evaluate our scoring algorithm for CAS queries.

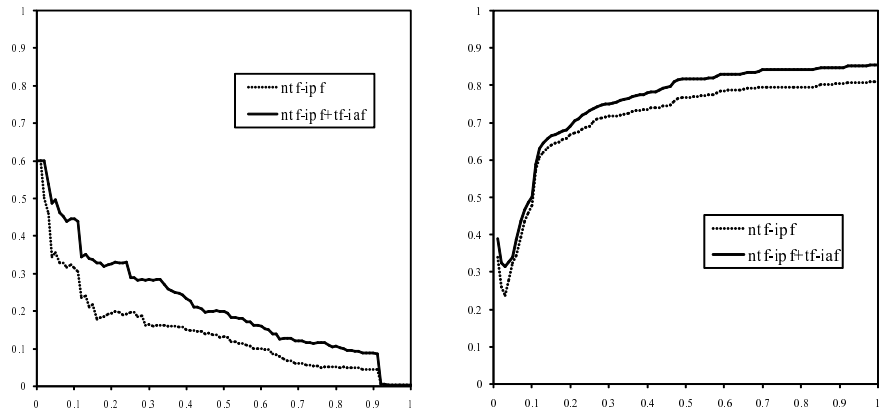

Fig. 5. Retrieval Accuracies based on Recall-Precision/nXCG

Fig. 5 shows the retrieval accuracies based on the recall-precision and the nXCG in the INEX evaluation measures. We found that our scoring algorithm (hereinafter called "tf-ipf+tf-iaf") could retrieve more relevant XML fragments than one where only document-base score is considered (hereinafter called "tf-ipf") in the *Thorough* strategy. In fact, we noticed that the relevant XML fragments tended to be higher on the list of each search result using `tf-ipf+tf-iaf`. This is because the XML fragments whose exhaustivity is large may be ranked lower than ones whose exhaustivity is small using only *tf-ipf* scoring. Using both *tf-ipf* and *tf-iaf* scorings, on the other hand, XML fragments whose exhaustivity and specificity are large make a point of being ranked higher in the search results. In short, introducing *tf-iaf* scoring reflects exhaustivity in the scores of

[11] Finally, $S_d(s)$ is weighted by the length of s. In short, the smaller the length of s is, the smaller $S_d(s)$ is.

answer XML fragments, and helps to improve the retrieval accuracies of XML search engines.

In summary, we have verified the effectiveness of the *tf-iaf* scoring for CAS queries, because it can retrieve more relevant XML fragments compared with the *tf-ipf* scoring.

5 Related Work

The application of information retrieval techniques in searching XML documents has become an area of research in recent years. Especially, the participants in the INEX project have proposed a lot of scoring proposals for XML search [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 weights either in the vector space model or the probabilistic model, has increased retrieval accuracy. However, document statistics query conditions have 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 [23]. However, although XIRQL, their proposed language, enables queries with a mix of conditions on both structure and keywords, only keywords are scored using conditions on document structure. Other scoring methods also use conditions on document structure to apply length normalization between query paths and data paths [8], to compute term weights based on element tags or paths [6, 9], or to account for overlapping elements [13]. It was reported that these methods were useful for searching XML fragments [19]; however, such methods did not use statistics of original XML documents and structural conditions of queries.

We believe that we have to utilize everything extracted from XML documents and queries for searching XML fragments accurately. In this paper, therefore, we showed that accounting for document statistics and query structure in addition to the existing methods, and combining them to improve retrieval accuracies of XML search engines. We can verify the effectiveness of our above proposals through the experimental evaluation in Section 4.

6 Conclusion

XML is emerging as the standard format for presenting data and documents on the Internet, and XML search engines are becoming necessary. Existing XML search engines can consider the content and the structure of XML documents to rank answer XML fragments to the XML queries. However, XML queries combine conditions on content and structure of both document and queries. That is, depending on the types of XML queries, we have to use the *tf-ipf* and the *tf-iaf* scorings. Based on this consideration, we proposed a method of content- and structure-based scorings in the vector space model considering both document and query conditions. Our method integrates document- and structure-based term-weighting strategies for XML search. Using our method, we found that we

could retrieve more relevant XML fragments with higher retrieval accuracy than using conventional scoring methods. We also proposed the displaying method to improve the retrieval accuracies of XML search engines. Unfortunately, we could not verify the effectiveness of this approach in this paper; however, we think that displaying search results is closely related to improving retrieval accuracies of XML search engines from the standpoint of users. This fact has already noticed in human interface research area, so that we have to implement our approach to our XML search engine as early as possible.

Acknowledgments

This work was partly supported by Grant-in-Aid for Scientific Research on Priority Areas #19024058 of the Ministry of Education, Culture, Sports, Science and Technology (MEXT), and Core Research for Evolutional Science and Technology (CREST) program "New High-performance Information Processing Technology Supporting Information-oriented Society" of the Japan Science under and Technology Agency (JST).

References

1. Bray, T., Paoli, J., Sperberg-McQueen, M., Maler, E., Yergeau, F.: Extensible Markup Language (XML) 1.0 (Fourth Edition). `http://www.w3.org/TR/xml` (Sep. 2006) W3C Recommendation 16 August 2006, edited in place 29 September 2006.
2. Salton, G., Buckley, C.: Term-Weighting Approaches in Automatic Text Retrieval. Information Processing and Management **24**(5) (1988) 513–523
3. Trotman, A., Sigurbjörnsson, B.: Narrowed Extended XPath I (NEXI). In: Advances in XML Information Retrieval. Volume 3493 of Lecture Notes in Computer Science., Springer-Verlag (May 2005) 16–40
4. Amer-Yahia, S., Botev, C., Dorre, J., Shanmugasundaram, J.: XQuery Full-Text extensions explained. IBM Systems Journal **45**(2) (Dec. 2006) 335–352
5. Amer-Yahia, S., Lalmas, M.: XML Search: Languages, INEX and Scoring. SIGMOD Record **35**(4) (Dec. 2006) 16–23
6. Cohen, S., Mamou, J., Kanza, Y., Sagiv, Y.: XSEarch: A Semantic Search Engine for XML. In: Proceedings of 29th International Conference on Very Large Data Bases. (Sep. 2003) 45–56
7. Amer-Yahia, S., Koudas, N., Marian, A., Srivastava, D., Toman, D.: Structure and Content Scoring for XML. In: Proceedings of the 31st International Conference on Very Large Data Bases. (Aug./Sep. 2005) 361–372
8. Carmel, D., Maarek, Y.S., Mandelbrod, M., Mass, Y., Soffer, A.: Searching XML Documents via XML Fragments. In: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. (Jul./Aug. 2003) 151–158
9. Grabs, T., Schek, H.J.: PowerDB-XML: A Platform for Data-Centric and Document-Centric XML Processing. In: Proceedings of the First International XML Database Symposium. Volume 2824 of Lecture Notes on Computer Science., Springer (Sep. 2003) 100–117

10. Fujimoto, K., Shimizu, T., Terada, N., Hatano, K., Suzuki, Y., Amagasa, T., Kinutani, H., Yoshikawa, M.: An Implementation of High-Speed and High-Precision XML Information Retrieval System on Relational Databases. In: Advances in XML Information Retrieval and Evaluation. Volume 3977 of Lecture Notes in Computer Science., Springer (June 2006) 254–267
11. Hatano, K., Kinutani, H., Amagasa, T., Mori, Y., Yoshikawa, M., Uemura, S.: Analyzing the Properties of XML Fragments Decomposed from the INEX Document Collection. In: Advances in XML Information Retrieval. Volume 3493 of Lecture Notes in Computer Science., Springer (May 2005) 168–182
12. Hatano, K., Kinutani, H., Watanabe, M., Mori, Y., Yoshikawa, M., Uemura, S.: Keyword-based XML Fragment Retrieval: Experimental Evaluation based on INEX 2003 Relevance Assessments. In: Proceedings of the 2nd Workshop of the Initiative for the Evaluation of XML Retrieval. (March 2004) 81–88
13. Clarke, C.L.A.: Controlling Overlap in Content-Oriented XML Retrieval. In: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. (Aug. 2005) 314–321
14. Yoshikawa, M., Amagasa, T., Shimura, T., Uemura, S.: XRel: A Path-based Approach to Storage and Retrieval of XML Documents using Relational Databases. ACM Transactions on Internet Technology **1**(1) (Aug. 2001) 110–141
15. Salton, G., Wong, A., Yang, C.S.: A Vector Space Model for Automatic Indexing. Communication of the ACM **18**(11) (Nov. 1975) 613–620
16. Clark, J., DeRose, S.: XML PAth Language (XPath) Version 1.0. `http://www.w3.org/TR/xpath` (Nov. 1999) W3C Recommendation 16 November 1999.
17. Liu, F., Yu, C.T., Meng, W., Chowdhury, A.: Effective Keyword Search in Relational Databases. In: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, ACM (June 2006) 563–574
18. Shimizu, T., Yoshikawa, M.: XML Information Retrieval Considering Physical Page Layout of Logical Elements. In: Proceedings of the 10th International Workshop on Web and Databases. (June 2007) 48–49
19. Kazai, G., Lalmas, M.: INEX 2005 Evaluation Metrics. In: Advances in XML Information Retrieval and Evaluation. Volume 3977 of Lecture Notes on Computer Science., Springer-Verlag (Jun. 2006) 16–29
20. Kekäläinen, J., Järvelin, K.: Using Graded Relevance Assessments in IR Evaluation. Journal of the American Society for Information Science and Technology **53**(13) (Nov. 2002) 1120–1129
21. Kazai, G., Lalmas, M.: eXtended Cumulated Gain Measures for the Evaluation of Content-Oriented XML Retrieval. ACM Transactions on Information Systems **24**(4) (Oct. 2006) 503–542
22. Kazai, G., Lalmas, M., de Vries, A.P.: The Overlap Problem in Content-Oriented XML Retrieval Evaluation. In: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. (Jul. 2004) 72–79
23. Fuhr, N., Großjohann, K.: XIRQL: An XML Query Language based on Information Retrieval Concepts. ACM Transactions on Information Systems **22**(2) (Apr. 2004) 313–356