Result Reconstruction Approach for More Effective XML Fragment Search

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

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

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

ABSTRACT

We propose and evaluate a method for obtaining more accurate search results in extensible markup language (XML) fragment search, which is a search that produces only relevant fragments or portions of an XML document. The existing approaches generate a ranked list in descending order of each XML fragment's relevance to a search query; however, these approaches often extract irrelevant XML fragments and overlook more relevant fragments. To address these problems, our approach extracts relevant XML fragments by considering the size of the fragments and the relationships between the fragments. Next, we score the XML fragments to generate a refined ranked list. For scoring, we rank the XML fragments that are informative for user information needs as high in the list. In particular, each XML fragment is scored using the statistics of its descendant and ancestor XML fragments.

Our experimental evaluations show that the proposed method outperforms BM25E, a conventional approach, which neither reconstructs XML fragments nor uses descendant and ancestor statistics.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

PERFORMANCE

Keywords

XML fragment search, integration and refinement of XML fragments, statistics of XML fragments

1. INTRODUCTION

The extensible markup language (XML) is a markup language for structured documents that has become the de facto format for data exchange. A large number of XML documents are available on the Web. We expect this trend to continue in the future. As such, information retrieval techniques for searching XML documents are very important and required in the field.

An XML fragment is usually defined as a part of a larger XML document. The fragment is identified by the surrounding start and end tags. XML fragments are, therefore, nested and have containment relationships with each other, which can cause overlaps in XML fragments in an XML document. The key goal of XML search is to obtain relevant XML fragments for a query instead of just returning the entire XML documents. Therefore, XML search engines can generate a ranked list composed of a set of relevant XML fragments, while several Web search engines return a set of entire Web documents. By isolating the XML fragments, users do not need to identify the relevant XML fragments from the larger XML documents.

There are two key research areas regarding XML fragment search: (1) improving the efficiency of a query execution and (2) generating a more accurate ranked list of results. Since an XML document contains numerous XML fragments, there is a much larger number of search targets in XML fragment search as compared with the conventional document search. Therefore, the efficiency of query execution is critical. This issue is being addressed by existing approaches for efficient XML search, high-performance machines, and sophisticated database management systems. On the other hand, the latter problem of search result accuracy has been paid less attention, although it must also be studied in depth to survive in this era of information explosion.

Studies focusing on effective XML search have indeed been performed. As described above, the XML fragments in an XML document often overlap. Some existing approaches ignore the overlap problem and present XML fragments as a list in which these fragments are arranged in descending order of their score, as calculated during query processing[1]. We call this list, which does not remove overlapping results, a *simple ranked list*. Kazai et al.[6] found that a simple ranked

[1]Several search engines select the top k ranked XML fragments, where k depends on the given system.

iiWAS2010, 8-10 November, 2010, Paris, France.

list deteriorates the search accuracy when it is composed of overlapping XML fragments. Therefore, almost all studies on XML fragment search use a *non-overlapped ranked list* instead.

The INitiative for the Evaluation of XML retrieval (INEX) project[2], the largest project addressing XML search, also uses a non-overlapped ranked list for the ad hoc track whose goal is to attain effective search. To generate a non-overlapped ranked list, we must detect and omit overlaps between the XML fragments in the list.

Search results must be composed of the XML fragments that are informative enough and match user information needs; however, the existing approaches simply calculate a relevance score for each XML fragment in relation to a given query. This is not enough; although such a non-overlapped ranked list does not typically return completely irrelevant XML fragments, these approaches do not consider containment relationships between the fragments in an XML document, and therefore, accuracy suffers. For example, when an overlap occurs, an XML fragment with a lower score is extracted after another XML fragment with a higher score has been extracted. In this case, the latter XML fragment is a part of the former one, indicating that the former one may also be reasonable as a relevant XML fragment.

Based on the above example, we should judge which XML fragment is more appropriate, and therefore, we consider the containment relationships. We can further identify and improve the precision of XML fragments by using statistics derived from the descendant or ancestor XML fragments. Combining these approaches, we generate a *refined ranked list* from a simple ranked list and improve the accuracy and precision of our XML fragment results.

The rest of this paper is organized as follows. We introduce the related studies in Section 2, followed by a detailed explanation of our approach in Section 3. Section 4 presents our experimental results and offers a discussion on our findings. Finally, we conclude our paper in Section 5.

2. RELATED STUDIES

There are two types of XML documents: (1) data-centric which mainly contain single or compound term in their text nodes and (2) document-centric which tend to contain natural languages in their text nodes[1]. The following subsections describe the existing studies related to both types of XML documents. Although we are primarily interested in search techniques for document-centric XML documents, information on data-centric XML documents will also be useful.

2.1 Data-Centric XML

Data-centric XML documents generally describe only one term in their text nodes. Therefore, studies investigating data-centric XML primarily focus on searching query keywords. The existing research efforts to attain efficient XML fragment search usually utilize the lowest common ancestor (LCA) approach[15]. As a part of this approach, the LCA itself may originate as the top-level common ancestor of arbitrary nodes in an XML tree; however, it is generally defined as the deepest node containing all query keywords in its descendants. Research involving LCA and XML fragments shows significant results related to *efficient* XML search; however, such techniques do not perform well in the context of accurate XML search. In other words, the retrieval accuracy of XML search engines decreases when we use LCAs as the most appropriate XML fragments for a given query[11].

To address this problem, research efforts have also tried to identify and extract more relevant XML fragments from the sub-tree whose root node is an LCA. XSeek[11] is one such solution, producing a meaningful LCA (MLCA), which classifies and analyzes XML tags by using XML schema information and the positions of query keywords. In the case of XSeek, nodes related to a query are selected and extracted in the order of their relevance to a query. Another approach, eXtract[4], is an expansion of MLCA and infers a user's search purpose by analyzing queries. In eXtract, queries are classified into two cases: (1) extracting an explicit search target and (2) extracting neighbors of the search target.

The purpose of these approaches is similar to that of ours: we wish to return the reconstructed XML fragments. Our method is, however, based on an information retrieval technique for achieving effective XML search, while the above-mentioned approaches utilize LCA for the purpose of efficiency.

2.2 Document-Centric XML

Since most document-centric XML documents contain multiple terms in their text nodes, the existing approaches for searching document-centric XML documents focuses on an effective search. In other words, the key objective is to rank more relevant XML fragments higher in the result list. Therefore, conventional information retrieval techniques are often utilized. Another approach is to refine the result list, for example, by removing insignificant fragments or reducing their scores.

Scoring methods for XML fragment search are often derived from the ones used in document search. For example, TF-IPF[2], a popular scoring method for XML fragment search, is an XPath[3]-based scoring method that extends the well-known TF-IDF[14] approach for document search. Another popular approach for XML fragment search is BM25E[10], which is based on Okapi's BM25[13] scoring method for document search.

Although these approaches have been successful, a gap between XML fragment search and document search remains. The goal of XML fragment search is to extract the relevant part of an XML document directly, whereas that of document search is to simply find relevant documents. Extracting relevant XML fragments for a given query is similar to generating snippets in a traditional document search. Snippets, which are short summaries of Web pages that help users judge whether the documents are worth reading, are generated by using information extraction techniques. Since both XML fragments and snippets present useful information to users, such information extraction techniques can be applied to our scoring method.

Manning et al. noted that snippets should have useful information and be maximally informative to a query[12]. To satisfy such requirements, we consider statistics calculated by query conditions[7]; such statistics consist of two components: (1) the ratio of XML fragments containing query keywords amongst XML fragments satisfying the constraints of the structure of the given query and (2) the number of query keywords in each XML fragment. In this paper, we

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

[3] http://www.w3c.org/TR/xpath

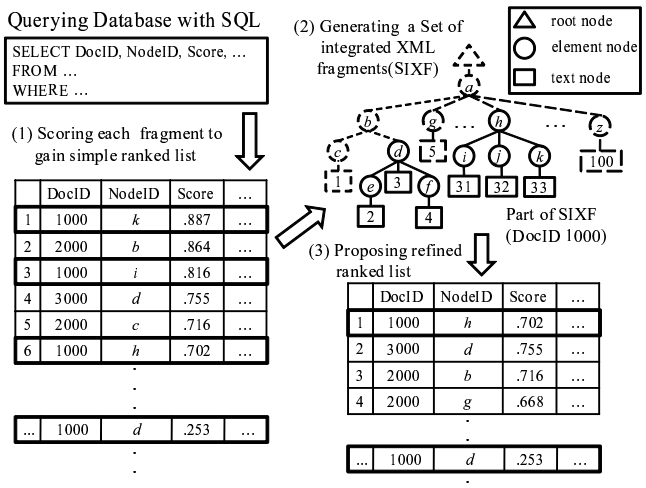


Figure 1: Overview of our proposed method

denote (1) as the query structure score and (2) as the query keyword score. In our previous study[7], [8], our experiments showed that such methods are more effective than the approaches that do not consider query statistics. We also found that the length of the retrieved XML fragments increases when we use these statistics.

As mentioned above, removing useless or low-scoring fragments is also effective. Although all granularities of XML fragments should be treated as search targets, the effectiveness of the results decreases sharply if a search engine returns noninformative XML fragments. Extremely small XML fragments are often not suitable for search results; Hatano et al. noted that when such meaningless XML fragments are removed, search accuracy improves[3]. Furthermore, our previous study suggests that large XML fragments are also inappropriate for search results[8]. Therefore, we should consider the length of XML fragments.

3. SEACHING FOR XML FRAGMENTS

Because users tend to browse only the top search results, we believe it is crucial to attain highly accurate top-ranked results. Similarly, INEX evaluates the precision of the relevant XML fragments at a lower recall level. Search engines should return the XML fragments that are as small as possible for the given query; however, recently, document search has been regarded as more effective than XML fragment search (as demonstrated by INEX). We believe that XML fragment search enables users to save time and energy in their information retrieval tasks and could be very convenient. Therefore, we aim to propose a method to find XML fragment search results whose length is appropriate.

In our approach, we show the *refined ranked list*, which is composed of relevant and informative XML fragments. As shown in Figure1, the overview of our proposed method is as follows:

(1) We first score each XML fragment by using a scoring method to obtain a simple ranked list.

(2) We extract XML fragments from a simple ranked list and generate a set of relevant XML fragments by considering limitations of the XML fragment length and reconstruction of XML fragments.

(3) To present a refined ranked list, we rank the XML fragments extracted from Step (2); we re-rank and remove the XML fragments in the simple ranked list and incorporate them into the refined ranked list.

Hereafter, we denote the set of relevant XML fragments generated in Step (2) as the *Set of Integrated XML Fragments* (*SIXF*). Furthermore, to more readily explain the process of generating the SIXF, we translate each XML fragment into a tree structure. In general, XML documents can be translated into a tree structure by representing tags as nodes in the tree. Each XML fragment in an XML document is associated with a sub-tree in the entire XML tree of the XML document.

3.1 Selecting an Effective Scoring Method

Our method starts with a simple ranked list. Therefore, the search accuracy of the proposed method depends on the quality of the given list. Selecting a suitable scoring method for an XML fragment search is the key to acquiring a reliable simple ranked list.

We assume the following two conditions:

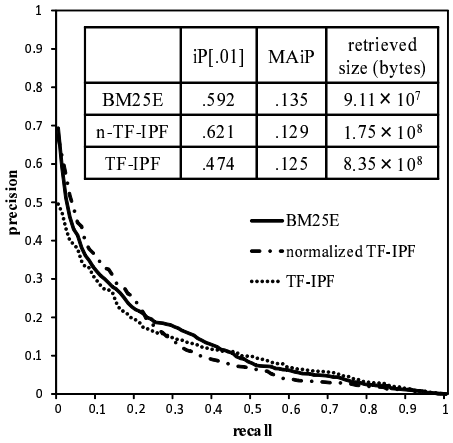
- A simple ranked list is highly accurate in terms of its mean average interpolated precision (MAiP)[4]
- A simple ranked list contains XML fragments of varying lengths.

Since our proposed method transforms a simple ranked list into a more effective one, we need to first obtain as many relevant XML fragments as possible. Therefore, we need to utilize a method with high MAiP.

Kamps et al. noted that large-sized XML fragments tend to be more effective[5] compared with middle-sized or small-sized ones however, problems can occur while extracting such large-sized fragments. Consider an XML document in which several relevant fragments exist. Using the tree structure shown in Figure 1, we consider sub-trees with root nodes d and h to be relevant fragments. When we extract a sub-tree that contains all relevant fragments, it might also include numerous irrelevant fragments. For example, if we choose a sub-tree with root node a, it contains numerous irrelevant fragments including c and g. If the scoring method used gives a higher score to large-sized XML fragments, result a may rank higher than d and h. To solve this problem, we should extract multiple relevant XML fragments from the XML document. In particular, we need to extract variously sized XML fragments in order to determine the appropriate granularity of XML fragments. Given this approach, we focus on MAiP and the length of the retrieved XML fragments.

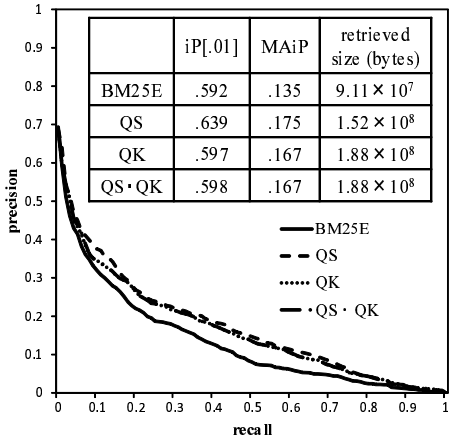
Based on the above discussion, we performed a preliminary experiment to determine an appropriate scoring method. In our experiment, we compared the search accuracy of the following three scoring methods (from the INEX 2008 test collection[5][5]): TF-IPF[2], normalized TF-IPF[9], and BM25E[10].

[4]MAiP is an evaluation measure that calculates the average of the (mean) interpolated precision at each recall level. The INEX assessment tool divides recall levels into 101 levals.
[5]Note that a detailed explanation of the INEX test collection is described in Section 4.1



	iP[.01]	MAiP	retrieved size (bytes)
BM25E	.592	.135	9.11×10^7
n-TF-IPF	.621	.129	1.75×10^8
TF-IPF	.474	.125	8.35×10^8

Figure 2: Comparison of scoring methods



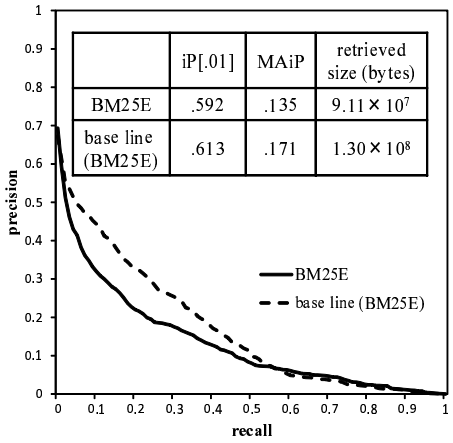
	iP[.01]	MAiP	retrieved size (bytes)
BM25E	.592	.135	9.11×10^7
QS	.639	.175	1.52×10^8
QK	.597	.167	1.88×10^8
QS・QK	.598	.167	1.88×10^8

Figure 3: Search accuracy of scoring methods of [7]

Since BM25E requires a weight to be assigned to each tag, we simply set the weight of all tags to 1. When we extract multiple XML fragments from a single XML document, we cannot determine whether a large XML fragment is extracted or several smaller XML fragments are extracted. To determine the best approach amongst the three alternatives, we extract only one XML fragment in an XML document.

As shown in Figure 2, we conclude that BM25E is the most suitable, because it has the highest MAiP and the smallest retrieved text size. Small retrieved text size means a ranked list extracted not only large-sized XML fragments but also extracted small-sized ones. Therefore, we utilize BM25E as the scoring method in our subsequent experimentation.

Next, we should demonstrate whether our assumption is correct—i.e., whether variously sized XML fragments are helpful. To verify this, we compare other scoring methods with BM25E. As discussed in Section 2.2, two types of statistics that we considered from [7] are useful for the effective search and tend to regard large XML fragments as more relevant. To confirm the effect of these scoring methods, we evaluated three methods: (1) query structure score, denoted as QS; (2) query keyword score, denoted as QK; and (3) the combination of both QS and QK.



	iP[.01]	MAiP	retrieved size (bytes)
BM25E	.592	.135	9.11×10^7
base line (BM25E)	.613	.171	1.30×10^8

Figure 4: Comparison of scoring methods in base line

As shown in Figure 3, the experimental results show that QS produces the highest MAiP value, as well as much larger XML fragments. Therefore, we use QS as our preferred search method.

3.2 Generating a Set of Integrated XML Fragments

To generate an SIXF, we extract the relevant XML fragments from the simple ranked list generated in Step (1) of our method. As discussed in Section 3.1, large XML fragments might contain irrelevant fragments and decrease the search accuracy. Therefore, we extract multiple relevant XML fragments from an XML document, as long as their sizes are properly restricted. Before discussing our reconstruction of XML fragments, we first confirm the potency of extracting multiple XML fragments from a single XML document.

One base line approach for generating a nonoverlapped ranked list of XML fragments is to repeatedly extract XML fragments from a simple ranked list in descending order of their rank unless an overlap occurs. The overlapped XML fragments are simply discarded and ignored.

This operation continues as long as either a candidate of the XML fragments remains in the search results or the number of the extracted XML fragments reaches a predefined upper limit[6].

We performed an experiment to verify the effectiveness of this base line approach. As shown in Figure 4, this base line approach increases the search accuracy of BM25E. We, therefore, conclude that extracting multiple XML fragments from an XML document is effective.

Next, we aim to reconfigure XML fragments in a simple ranked list to produce results that are better than those of the established base line. As noted in Section 2.2, we should consider how to identify and extract XML fragments of appropriate lengths in order to attain more accurate XML fragment search. We should also consider how to handle overlapping results, which we ignored in the base line approach.

[6]In our experimentation, we extracted 1,500 or less XML fragments for each query.

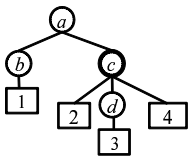


Figure 5: Example of *overwrite* fragments

From the above discussion, we derive the following requirements:

Requirement 1:
Since the traditional search results include several large XML fragments, we should impose an extraction limit on the fragment size.

Requirement 2:
The extracted XML fragments are appropriately abbreviated and reconstructed to resolve overlap problem.

3.2.1 Extraction Limit

To satisfy Requirement 1, we need to limit the size of the extracted XML fragments to an *extraction limit* (EL). Furthermore, we should summarize each XML document, because showing entire documents in search results is not effective. Therefore, we limit the extracted text size for each XML document by defining the EL of an XML document D, whose document ID ($DocID$) is as follows:

$$EL_{DocID} = \alpha \cdot |D_{DocID}| \quad (1)$$

where $|D_{DocID}|$ is the size of the XML document D_{DocID} and α is the ratio of the size of the relevant fragment.

Given this definition, we extract XML fragments from a simple ranked list when the text size of the XML fragments in the SIXF is less than EL. This process repeats until its size exceeds EL.

3.2.2 Reconstructing Fragments

To satisfy Requirement 2, we need to arrange the extracted XML fragments so that the SIXF contains useful search results. For generating a non-overlapped ranked list for the base line approach, we simply eliminate the overlapping XML fragments. This may prevent us from extracting relevant XML fragments. For example, in Figure 5, we assume that the XML fragment rooted at node c is the most relevant one in the tree; however, we cannot extract c if we have already extracted d.

To address this problem, we search larger XML fragments and *overwrite* them. As a result, these relevant fragments are all contained in the SIXF, while the existing approaches extract the fragments with the higher score. As these *overwrite* operations are applied, XML fragment lengths in the SIXF increase; therefore, the *overwrite* operation is executed only when Requirement 1 is satisfied.

Again, consider Figure 5. Suppose that c has been extracted. If a is extracted, a overwrites c, because it is larger than c. Furthermore, we discard d, because it is smaller than c.

3.3 Generating a Refined Ranked List

After generating a SIXF, we score each XML fragment in the SIXF and finally generate a refined ranked list. In this process, we aim to obtain informative XML fragments for the higher ranked results. The simplest way to score these XML fragments is to use BM25E, which is used in Step (1) of our method. Unless noted otherwise, we use the BM25E score in the remainder of this paper.

In the following subsections, we propose two scoring methods for generating a refined ranked list: (1) *bottom-up* scoring, which utilizes statistics of descendant XML fragments in order to score an ancestor XML fragment; and (2) *top-down* scoring, which utilizes statistics of an ancestor XML fragment in order to score descendant XML fragments.

3.3.1 Bottom-Up Scoring

The orgit overwrite operation introduced in Section 3.2.2 is executed when overlaps exist in the SIXF. In the *overwrite* operation, an ancestor XML fragment is extracted in place of its descendants. The ancestor should be ranked lower in a simple ranked list as compared to its descendants, implying that the refined ranked list also treats the ancestor XML fragment as a lower rank if BM25E is used. As a result, the originally higher-ranked XML fragments cannot always be ranked higher. To score these XML fragments more appropriately, we propose *bottom-up* scoring in order to give more reasonable scores to the XML fragments that contain highly scored descendants.

When we score an XML fragment, we should consider the statistics of its descendant fragments. Conversely, it is not appropriate that XML fragments with low scores are ranked high. Therefore, we must integrate these scores properly. Portions of text nodes in an ancestor XML fragment are composed of descendant XML fragments; the scores of the text nodes in these descendants affect those in the ancestors. Therefore, the *bottom-up* score should be calculated using the BM25E score, as well as the ratio of the lengths of the ancestor XML fragment and that of its descendants.

Let f_a be an ancestor XML fragment and f_d be the descendant fragment with the highest score. We define the *bottom-up* (*BU*) scoring function as

$$s_{BU}(f_a) = \frac{|f_d|}{|f_a|} \cdot s(f_d) + \frac{|f_a| - |f_d|}{|f_a|} \cdot s(f_a) \quad (2)$$

where $|f_d|$ is the length of f_d, $|f_a|$ is the length of f_a, $s_{BU}(f_a)$ is the *bottom-up* score of f_a, $s(f_a)$ is the BM25E score of f_a, and $s(f_d)$ is the BM25E score of f_d.

3.3.2 Top-Down Scoring

Since query keywords may have numerous meanings, it is often difficult to identify a proper one. One solution is to consider the co-occurrence of query keywords. If an XML fragment contains several distinct query keywords in its text nodes, we can assume the XML fragment to be closely related to the meaning of the given query keywords. In our previous study[7], we obtained informative XML fragments by considering the number of distinct query keywords in each XML fragment.

The larger XML fragments contain more query keywords, indicating that larger XML fragments tend to be ranked higher. In other words, we might overlook smaller XML fragments, although they are informative. To cope with this problem, we propose a scoring method that is independent of XML fragment size.

XML fragments contain numerous distinct query keywords and are identified as informative. Descendant XML fragments of these informative fragments should also be informative. We, therefore, consider *top-down* scoring by calculating the ratio of the number of distinct query keywords contained in an XML fragment to that of its top-level ancestor—i.e., the entire document.

Let f be a scored XML fragment and D_f be an XML document associated with f. We define the *top-down* (*TD*) scoring function as

$$s_{TD}(f) = s(f) \cdot count(D_f) \quad (3)$$

where $s(f)$ is the BM25E score of f and $count(D_f)$ is the number of distinct query keywords in D_f.

These two methods, *BU* and *TD* can be integrated. We call mixture scoring method as *BU·TD*.

3.4 Example of Generating SIXF and Refined Ranked List

In summary, we illustrate an example of generating a part of an SIXF in which the document ID is 1000 and its corresponding refined ranked list uses *BU* scoring. Figure 6 provides a graphical view of this example.

Suppose that $\alpha = \frac{1}{3}$, and $|D_{1000}| = 300$. Then, $EL_{1000} = \alpha \cdot |D_{1000}| = 100$. Next, we introduce τ_{1000}, which is the total length of the extracted XML fragments from document ID 1000.

We first obtain a simple ranked list calculated in Step (1). The obtained XML fragments are shown in the left table of Figure 6. For the sake of simplicity, we assume that the list contains only the XML fragments whose document ID is 1000.

We extract the XML fragment with the highest score, which is node k, from the simple ranked list. Since the text length of k is 40 ($< EL_{1000}$), k is extracted. This extraction process continues because τ_{1000}, which contains text node 33, is less than EL_{1000}. Therefore, i is selected next, because i has the second-highest score in the simple ranked list. Thus, i is extracted because τ_{1000}, which contains text nodes 31 and 33, equals 50 ($< EL_{1000}$). Node h is the next candidate to be extracted. Since i and k are the descendants of h, they are *overwritten* by h. In particular, i and k are removed from the SIXF and h is added. At this point, τ_{1000} becomes 70, which is still less than EL_{1000}.

Next, *bottom-up* scoring is applied. Function (2) is used to score node $h(s_{BU}(h)) = \frac{40}{70} \cdot 0.887 + \frac{70-30}{70} \cdot 0.702 = 0.808$.

Following Figure 6 further, d is also extracted. Nodes b and c fail to be extracted because τ_{1000} exceeds EL_{1000}. In the end, the SIXF is formed by nodes d and h.

Finally, the refined ranked list is constructed by adding the scores calculated via *bottom-up* scoring into the SIXF. In the same manner, we generate complete SIXF for all XML documents and construct the final refined ranked list in the descending order of their scores.

4. EXPERIMENTAL EVALUATION

4.1 Test Collection

We performed our experimental evaluation by using the INEX 2008 test collection[5]. While it is not the latest one, we opted not to use the INEX 2009 test collection because

- There are various XML tags derived from YAGO[16] in INEX 2009. These tags help INEX participants to recognize the semantic knowledge of XML fragments in XML fragment search; however, we do not incorporate such semantics in our solution.

- We utilize the logical structure of XML documents in our approach. The XML documents of INEX 2009 tend to have overly complex and nested structures formed by the YAGO tags. This feature hinders us from exploiting the structure of the XML documents.

The INEX 2008 test collection is a Wikipedia XML Corpus based on a snapshot of the English Wikipedia in early 2006. It consists of three components: (1) the INEX document collection, (2) INEX topics, and (3) INEX relevance assessments. The INEX document collection contains 660,000 XML documents and the INEX topics include 68 queries. We use all topics in our experimental evaluation except for the experiment described in Section 4.4. There are two types of queries: (1) content-and-structure (CAS) queries, which constrain both structure and keywords; and (2) content-only (CO) queries, which constrain only keywords. Each query is represented as narrowed-extended XPath I (NEXI)[17].

The INEX relevance assessments are evaluation tools for XML fragment search. Using INEX relevance assessments, an XML search engine can be evaluated on the basis of some evaluation measures by inputting a ranked list into the INEX relevance assessments. We use this in our experimental evaluation. Although the official measure for the focused task in ad hoc track is iP[0.01], we also show MAiP to reveal the overall effectiveness of our proposed method.

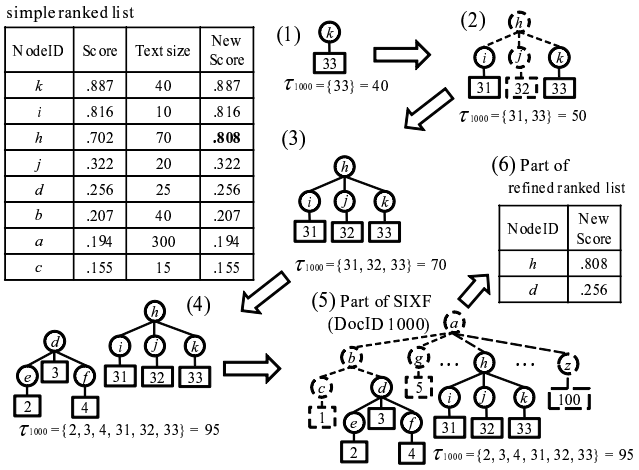
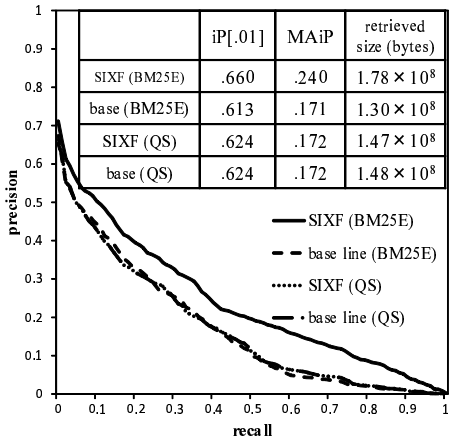


Figure 6: Example of generating a refined ranked list for a single XML document

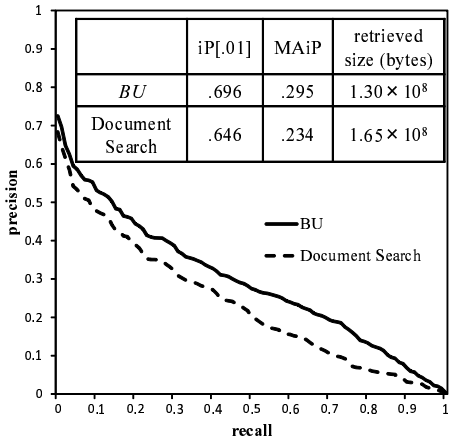
4.2 Experimental Evaluation of SIXF

We conducted experiments to compare our method, SIXF, with the aforementioned base line approach. Figure 7 shows that all interpolated precisions at each recall level of SIXF (BM25E) are higher than those of the base line (BM25E). iP[.01] of SIXF (BM25E) is improved by 7% as compared with that of the base line (BM25E). As a result, we conclude that the reconstruction of XML fragments is much effective for XML fragment search.



	iP[.01]	MAiP	retrieved size (bytes)
SIXF (BM25E)	.660	.240	1.78×10^8
base (BM25E)	.613	.171	1.30×10^8
SIXF (QS)	.624	.172	1.47×10^8
base (QS)	.624	.172	1.48×10^8

Figure 7: Comparison of our reconstruction method versus the base line



	iP[.01]	MAiP	retrieved size (bytes)
BU	.696	.295	1.30×10^8
Document Search	.646	.234	1.65×10^8

Figure 8: Comparison of XML fragment search and document search

As shown in Figure 7, the accuracy of SIXF (QS) is almost equal to the corresponding base line (QS). The reconstruction of XML fragments with QS is not effective. BM25E gives high scores to variously sized XML fragments, whereas QS tends to give high scores to larger ones. This suggested that a scoring method that can return variously sized XML fragments is suitable for our goal, because such a scoring method can return a more appropriate granularity of XML fragments when an *overwrite* operation occurs. We, therefore, conclude that the length of the XML fragments in a simple ranked list affects the search accuracy of SIXF.

Furthermore, our experiments show that *EL*, introduced in Section 3.2.1, does not perform well. As a consequence, we do not apply this approach to the experiments in the remaining subsections.

4.3 Effectiveness of Scoring Methods for a Refined Ranked List

We evaluated the effectiveness of *BU* and *TD* scoring methods in Section 3.3. As summarized in Table 1, both these methods, as well as a mixture of the two methods (*BU·TD*) improved search accuracy versus SIXF (BM25E).

Note that *TD* and *BU·TD* increased the size of the retrieved XML fragments in a ranked list. This is an unpleasant result, because we believe that XML fragment search should produce search results that are small rather than large.

Conversely, *BU* decreased the size of the retrieved XML fragments. Therefore, *BU* is the most suitable method for our proposal. Finally, iP[.01] of proposed method is improved by 8% as compared with that of the base line. XML fragments are re-scored and re-ranked with a higher ranking when an *overwrite* operation occurs, indicating that we could abbreviate XML fragments as an appropriate granularity level because the retrieved text sizes are decreased.

Next, we compared with other systems of INEX participant. Table 2 compares our proposed method (*BU*) with three other competitive XML search engines of INEX participants in [5]. Only four systems (including ours) were evaluated using both CAS and CO queries. In the experiments, our proposed method provides the highest precision when the recall level is less than or equal to iP[.01]. Because the official measure for the focused task is iP[.01], our system has the highest level of accuracy.

4.4 Comparisons to Document Search

We also compare our proposed method (*BU*) with the traditional document search, because document search is often noted as being more effective than XML fragment search[5]. In the document search, the entire XML documents are returned as search results from the simple ranked list generated by BM25E. In this experiment, we used the 41 queries that return the entire XML documents.

As shown in Figure 8, the results of our experimentation showed that all interpolated precisions at each recall level of the proposed method are higher than those of the document search. This indicates that XML fragment search is more effective than document search. Therefore, we conclude that it is reasonable to limit the length of the returned XML fragments, although the approach in Section 3.2.1 did not perform well.

Furthermore, we note that the retrieved text size of our proposed method is substantially smaller than that of document search. As such, we present more focused content to users, which saves their time and energy.

5. CONCLUSION

In this paper, we proposed a method to improve the effectiveness of XML fragment search. We attempt to identify XML fragments that are relevant to a given query. We score only these XML fragments to generate a more accurate ranked list.

The experimental results show that our proposed method, *BU*, is more effective than the traditional approaches, one of which we used as a comparative base line. We found that the accuracy of XML fragment search can be improved by reconstructing XML fragments and emphasizing on informative ones by applying statistics of descendant XML fragments. We also reduced the text size of the retrieved XML fragments, implying that users do not have to sift through larger fragments to find the information they are searching for. Furthermore, we found that the search accuracy of our proposed method depends on the scoring function used to generate an initial simple ranked list. In our experiments, BM25E is the most effective scoring method, because it gives

Table 1: Effect of scoring methods on SIXF

	BU	*BU·TD*	*TD*	SIXF(BM25E)	base line (BM25E)
iP[.01]	.664	.664	.662	.660	.613
MAiP	.238	.239	.242	.240	.171
retrieved size (bytes)	1.75Ą 10^8	2.31Ą 10^8	2.28Ą 10^8	1.78Ą 10^8	1.30Ą 10^8

Table 2: Comparison of four INEX participant systems including our proposed system

Team	iP[.00]	iP[.01]	iP[.05]	iP[.10]	MAiP
Doshisha Univ.	.7045	.6639	.5510	.4949	.2379
Renmin Univ. of China	.5969	.5969	.5815	.5439	.2486
Queensland Univ. of Technology	.6232	.6220	.5521	.4617	.2134
Univ. of Amsterdam	.6514	.6379	.5901	.5280	.2261

high scores to variously sized XML fragments.

One of our future challenges is to further limit the text size of the returned XML fragments in order to achieve a more effective XML fragment search.

Acknowledgements

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

6. REFERENCES

[1] H. Blanken, T. Grabs, H. Schek, R. Schenkel, and G. Weikum. *Intelligent Search on XML Data: Applications, Languages, Models, Implementations, and Benchmarks*. Lecture Notes on Computer Science. Springer-Verlag, September 2003.

[2] T. Grabs and H. Schek. PowerDB-XML: A Platform for Data-Centric and Document-Centric XML Processing. In *Proc of the First International XML Database Symposium*, Lecture Notes on Computer Science, pages 100–117. Springer Berlin, September 2003.

[3] K. Hatano, H. Kinutani, M. Watanabe, Y. Mori, M. Yoshikawa, and S. Uemura. Keyword-based XML Portion Retrieval: Experimental Evaluation based on INEX 2003 Relevance Assessments. In *Proc of the Second Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 81–88, March 2004.

[4] Y. Huang, Z. Liu, and Y. Chen. Query Biased Snippet Generation in XML Search. In *Proc of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 315–326. ACM, June/July 2008.

[5] J. Kamps, S. Geva, A. Trotman, A. Woodley, and M. Koolen. Overview of the INEX 2008 Ad Hoc Track. In *INEX 2008 Workshop Pre-proceedings*, pages 1–28, December 2008.

[6] G. Kazai, M. Lalmas, and A. Vries. The Overlap Problem in Content-Oriented XML Retrieval Evaluation. In *Proc of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 72–79, July 2004.

[7] A. Keyaki, K. Hatano, and J. Miyazaki. A Query-oriented XML Fragment Search Approach on A Relational Database System. *Journal of Digital Information Management (JDIM)*, 8(3):175–180, June 2010.

[8] A. Keyaki, J. Miyazaki, and K. Hatano. A Method of Generating Answer XML Fragment from Ranked Results. In *INEX 2009 Workshop Pre-Proceedings*, pages 563–574, December 2009.

[9] F. Liu, C. Yu, W. Meng, and A. Chowdhury. Effective Keyword search in Relational Databases. In *Proc of the 2006 ACM SIGMOD international Conference on Management of Data*, pages 563–574. ACM, June 2006.

[10] W. Liu, S. Robertson, and A. Macfarlane. Field-Weighted XML Retrieval Based on BM25. In *Advances in XML Information Retrieval and Evaluation*, Lecture Notes on Computer Science, pages 161–171. Springer Berlin, June 2006.

[11] Ziyang Liu and Yi Chen. Identifying Meaningful Return Information for XML Keyword Search. In *Proc of the 2007 ACM SIGMOD International Conference on Management of Data*, pages 329–340. ACM, June 2007.

[12] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze. *Introduction to Information Retrieval*, pages 157–159. Cambridge University Press, July 2008.

[13] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. Okapi at TREC-3. *The Third Text REtrieval Conference (TREC-3)*, pages 109–126, 1995.

[14] G. Salton and C. Buckley. Term-Weighting Approaches in Automatic Text Retrieval. *Journal of Information Processing and Management*, 24(5):513–523, January 1988.

[15] A. Schmidt, M. Kersten, and M. Windhouwer. Querying XML Documents Made Easy: Nearest Concept Queries. In *Proc of the 17th International Conference on Data Engineering*, page 321. IEEE, April 2001.

[16] F. Suchanek, G. Kasneci, and G. Weikum. YAGOĄFA Core of Semantic Knowledge Unifying WordNet and Wikipedia. In *Proc of the 16th International Conference on World Wide Web (WWW2007)*, pages 697–706, May 2007.

[17] A. Trotman and B. Sigurbjörnsson. Narrowed Extended XPath I (NEXI). In *Advances in XML Information Retrieval*, pages 16–40, May 2005.