Analyzing the Properties of XML Fragments Decomposed from the INEX Document Collection

Kenji Hatano[1], Hiroko Kinutani[2], Toshiyuki Amagasa[1],Yasuhiro Mori[3], Masatoshi Yoshikawa[3], and Shunsuke Uemura[1]

[1] Nara Institute of Science and Technology, Japan
[2] Ochanomizu University, Japan
[3] Nagoya University, Japan

Abstract. In current keyword-based XML fragment retrieval systems, various granules of XML fragments are returned as retrieval results. The number of the XML fragments is huge, so this adversely affects the index construction time and query processing time of the XML fragment retrieval systems if they cannot extract only the answer XML fragments with certainty. In this paper, we propose a method for determining XML fragments that are appropriate in keyword-based XML fragment retrieval. This would help to improve overall performance of XML fragment retrieval systems. The proposed method utilizes and analyzes statistical information of XML fragments based on a technique of the dynamics of terminology in quantitative linguistics. Moreover, our keyword-based XML fragment retrieval system runs on a relational database system. In this paper, we briefly explain the implementation of our system.

1 Introduction

Extensible Markup Language (XML) [5] is becoming widely used as a standard document format in many application domains. In near future, a great variety of documents will be produced in XML. Therefore, in a similar way to the development of Web search engines, XML information retrieval systems will become very important tools for users wishing to explore XML documents.

In the research area of XML retrieval, it is important to develop a method for retrieving fragments of XML documents. XQuery [4], proposed by the World Wide Web Consortium (W3C), is known as a standard query language for XML fragment retrieval. Using XQuery, users can issue a flexible query consisting of both keywords and XPath notations. If users already have knowledge of the structure of XML documents, users can issue XQuery-style queries for XML fragment retrieval. Consequently, we can state that XQuery is suitable for searching for data in XML documents[1].

[1] In this paper, we refer to this type of XML documents as *data-centric* XML documents.

N. Fuhr et al. (Eds.): INEX 2004, LNCS 3493, pp. 168–182, 2005.

At the same time, the XML Query Working Group has been developing powerful full-text search functions [3, 2] for XQuery. This is because there are many *document-centric* XML documents like articles in XML form, including structured information such as the names of authors, date of publication, sections, and sub-sections, as well as unstructured information such as the text content of the articles. However, document-centric XML documents like these have different XML schemas in each digital library, making it impossible to comprehend the structure of XML documents or to issue a formulated query like XQuery into XML fragment retrieval systems. Therefore, XML information retrieval systems should employ a much simpler form of query such as keyword search services without utilizing XQuery-style queries. Keyword search services enable users to retrieve needed information by providing a simple interface to information retrieval systems. In short, it is the most popular information retrieval method because users need to know neither a query language nor the structure of XML documents.

Considering the above background of XML fragment retrieval, it is not surprising that much attention has recently been paid to developing a keyword-based XML fragment retrieval system. Such systems usually decompose document-centric XML documents into XML fragments by using their markup and then generate an index of decomposed fragments for searching. In spite of their systems' simple approach to XML fragment retrieval, this method enables the user to retrieve XML fragments related to keyword-based queries fairly well. However, XML documents are decomposed as much as possible by using their markup; thus index construction time and query processing time are too long compared with current document retrieval systems. This is because returning various granules and the huge number of XML fragments as retrieval results adversely affects processing time unless XML fragment retrieval systems can extract only the answer XML fragments with certainty.

We believe that the XML fragments required for keyword-based fragment retrieval make up only a part of the decomposed fragments from document-centric XML documents. In short, there is a certain type of XML fragments that are never returned as retrieval results regardless of the issued keyword-based queries. In particular, extremely small XML fragments are unlikely to become retrieval results of keyword-based queries from the viewpoint of information retrieval research. Therefore, we could perform XML fragment retrieval more efficiently than with current systems if we could eliminate inappropriate fragments in XML fragment retrieval from the index file. To cope with this problem, we have to determine which XML fragments are appropriate in the XML fragment retrieval extracted from document-centric XML documents.

In this paper, we propose a method for determining the appropriate XML fragments needed to efficiently search XML fragments. Our method utilizes and analyzes statistical information of XML fragments decomposed from original documents based on a technique of the dynamics of terminology in quantitative linguistics. Our proposal holds the promise of not only reducing index construction time and query processing time of XML fragment retrieval systems but also

dealing with many types of document-centric XML documents, since statistical information does not depend on the structures of XML documents. We also perform some experiments to verify the effectiveness of our proposal.

2 Research Issues

There are two main types of keyword-based XML fragment retrieval systems. In this paper, we refer to these as *data-centric type* and *document-centric type* for convenience. The former is based on structured or semi-structured database systems with keyword proximity search functions that are modeled as labeled graphs, where the edges correspond to the relationship between an element and a sub-element and to `IDREF` pointers [1, 13, 16]. Dealing with XML documents as XML graphs facilitates the development of keyword-based information retrieval systems that are able to perform the retrieval processing efficiently. The latter type has been developed in the research area of information retrieval [9, 12], and it enables us to retrieve XML fragments without indicating the element names of XML documents. The major difference between these two types of XML fragment retrieval systems is in the data characteristics of their retrieval targets. In short, we assume that the former type focuses mainly on XML documents that have a data-centric view, whereas the latter type deals with those having a document-centric view. At the same time, almost all XML fragment retrieval systems currently assume the existence of the DTD of XML documents in either field of research. It is true that DTD enhances the retrieval accuracy and query processing time of their systems. However, there are some problems associated with searching heterogeneous XML fragments on the Web. Thus, other types of XML retrieval systems that do not utilize DTD are required. Consequently, XML fragment retrieval systems in the future will have to deal with heterogeneous XML documents whose structures are not uniform.

To meet the needs of the new architectures for XML fragment retrieval systems, we have been developing a keyword-based XML fragment retrieval system [15]. Our system focuses on retrieval of document-centric XML documents rather than that of data-centric ones, and it does not utilize any information on elements of the XML documents, whereas almost all existing XML fragment retrieval systems take advantage of this information for querying and indexing XML documents. In our approach, XML documents must be decomposed into their fragments, and the decomposed fragments are utilized to generate an index file. XML is a markup language, thus XML documents can be automatically decomposed into their fragments by using their markup [19]. However, this gives rise to an unmanageable profusion of XML fragments. In other words, it takes a very long time to construct an index file and to search for XML fragments related to a keyword-based query. For this reason, it is critical to avoid inspecting all decomposed XML fragments, by focusing on the XML fragments that are appropriate to the XML fragment retrieval, in order to reduce index construction time and query processing time. In the next section, we explain the method for

determining the appropriate fragments in XML fragment retrieval based on a technique of the dynamics of terminology in quantitative linguistics.

3 Analysis of INEX Test Collection

Our research group has been analyzing the statistical information of the INEX document collection since last year. According to our analysis, it was notable that variances in the statistical information, especially the variance in the length of XML fragments, were too large. Therefore, we have focused on the length of XML fragments. In our INEX 2003 paper [14], we regarded small XML fragments as inappropriate ones in XML fragment retrieval and verified the practicality of our proposal by using recall-precision curves. However, we simply sketched an outline of our proposal and did not show strong reasons for adopting it. Consequently, in this section, we show the practical justification of our proposal.

3.1 Properties of XML Fragments

The Dynamics of Terminology in Quantitative Linguistics. In our INEX 2003 paper, we determined a threshold for the length of XML fragments, thus regarding XML fragments below this threshold as inappropriate for XML fragment retrieval. However, this approach required a large number of experiments to determine the threshold, so it was inappropriate for developing a large-scale XML fragment retrieval system. Therefore, we have to determine the threshold systematically.

It is well known that statistical information on XML fragments, such as the number of tokens, length of XML fragments, and so on, is useful for determining the thresholds. The examination of the relationships among the constituent elements and the type of conceptual combinations used in the construction of the terminology permits deep insights into the systematic thought processes underlying term creation. And the powerful interaction of linguistic possibilities and the limitation of conceptual entities are offered by the quantitative analysis of the patterns of the growth of terminology. In the dynamics of terminology in quantitative linguistics, statistical information is often used. This is because analyzing the statistical information helps us to discover some rules in a document set, and the discovery of such rules is essential for constructing a sound basis of a theory of terminology. In this research area, it is thought that conducting an examination to discover rules is similar to finding out the systematic processes underlying a document set. For this reason, we employ a technique of the dynamics of terminology to determine the threshold by using the number of tokens and XML fragments as statistical information[17]. Needless to say, not only the number of tokens and XML fragments but also other mathematical or algebraic information can be utilized as statistical information. The reason for using such statistical information is that it can be extracted easily when our XML fragment retrieval system simultaneously analyzes XML documents and decomposes them into fragments.

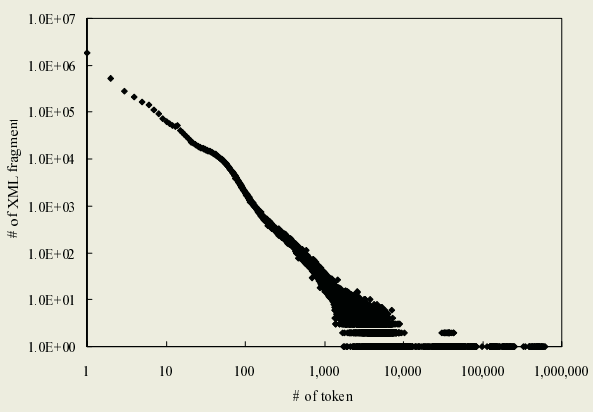

Fig. 1. Relationship between n and $N(n)$

By adopting a technique of quantitative linguistics, we can determine the threshold systematically.

Determining Appropriate XML Fragments. In the research area of quantitative linguistics, capturing properties of a document set is performed by analyzing statistical information. Utilizing the number of tokens and documents as statistical information, we can find a correlation between number of tokens and number of XML fragments with the same number of tokens. However, a small minority of documents have no relationship with the statistical information, so it is said that such documents have an anomalous property. Therefore, it is understood that such documents are not appropriate for capturing the properties of the document set and should be disregarded in capturing properties.

This concept can be utilized for determining inappropriate fragments in XML fragment retrieval. In short, if we are able to define a function between pieces of statistical information, XML fragments that do not follow the function can be regarded as inappropriate XML fragments. It is difficult to explain the process of determining inappropriate XML fragments on a conceptual basis, so we describe the process using the following example.

Figure 1 shows log-log plots of the relationship between the number of tokens and XML fragments of the INEX document collection, where n is the number of tokens in each XML fragment and $N(n)$ is the number of XML fragments that contain n tokens. This figure shows that the property of the INEX document collection is similar to that of Web document collection, since the log-log plots follow Zipf's distribution (or power-law distribution) [20]. Therefore, it is reasonable that statistics information of the INEX document collection follows Zipf's distribution. However, it is difficult to determine whether XML fragments, in general, follow Zipf's distribution.

From a statistical point of view in the dynamics of terminology, it can be said that the gaps between the plots in Figure 1 cause a harmful effect on statistical information. Therefore, statistical information in plots with gaps is not used

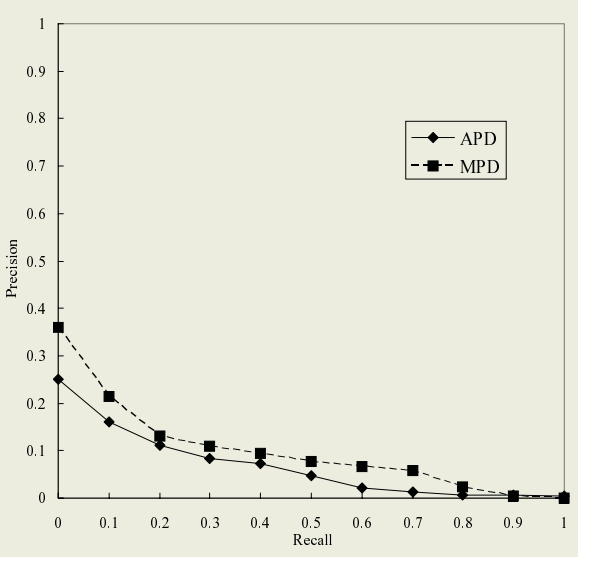

Fig. 2. Recall-precision curves based on the INEX 2003 relevance assessment

for capturing the property of the document set. In short, we consider the XML fragments in these plots with gaps to be inappropriate in XML fragment retrieval because we cannot capture the property of the document set accurately. As a result, we defined the appropriate fragments in XML fragment retrieval as the XML fragments in the plots in Figure 1 whose number of tokens is no smaller than 10 nor larger than 10,000 in the case of adopting the INEX document collection. This definition is sensible, because small XML fragments are not informative enough and large ones are too informative for users in keyword-based queries, and thus small/large XML fragments are unlikely to be answers to the CO-topics.

Verification of XML Fragments' Properties. In order to verify the validity of a technique from the dynamics of terminology, we performed some experiments using the INEX 2003 relevance assessment. In these experiments, we measured average precisions, index construction time, query processing time, and the number of indexed XML fragments of the following two types of index files in our XML fragment retrieval system: the index files of all XML fragments (APD) and those of the XML fragments other than the inappropriate fragments described in 3.1 (MPD).

Figure 2 shows the recall-precision curves based on the INEX 2003 relevance assessment. We initially expected that the recall-precision curves of APD and MPD would be very close to each other, since the fragments that were judged as inappropriate in XML fragment retrieval did not rank in the top 1,500 of all fragments. However, the recall-precision curve of MPD was higher than originally expected. Therefore, we think that the method proposed in 3.1 does not deteri-

Table 1. Comparison of APD with MPD in INEX 2003 relevance assessment

	# of fragments	index construction (s)
APD	8,224,053	513,878
MPD	1,011,202	109,115
	query processing (s/topic)	average precision
APD	17.66	0.0707
MPD	5.27	0.1038

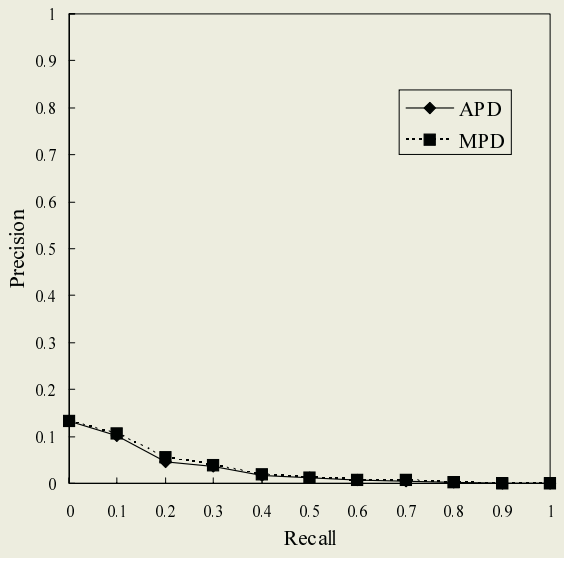

Fig. 3. Recall-precision curves based on the INEX 2004 relevance assessment

orate the retrieval accuracy of XML fragment retrieval systems. In addition, the number of indexed XML fragments was significantly reduced by adopting our method, thus also reducing index construction time and query processing time (see Table 1).

As shown above, the proposed method is useful not only in reducing index construction time and query processing time but also in improving the retrieval accuracy of an XML fragment retrieval system. Therefore, the proposed method is also suitable for the INEX 2004 relevance assessment.

3.2 Experiments Using INEX 2004 Relevance Assessment

Proposed method applied to the INEX 2003 relevance assessment worked well for reduction of index construction time and query processing time; thus we apply it to the INEX 2004 relevance assessment.

Figure 3 shows the recall-precision curves based on the INEX 2004 relevance assessment. Unlike the case of using the INEX 2003 relevance assessment, the recall-precision curves of APD and MPD were very close to each other. The

Table 2. Success ratios of our method (worst 5)

topic ID	$(E,S)=(2,3)$			$(E,S)=(3,2)$			$(E,S)=(3,3)$			total
	miss	all	success ratio	miss	all	success ratio	miss	all	success ratio	
187	378	554	0.32	1,028	1,463	0.30	646	848	0.25	0.50
166	9	15	0.40	5	48	0.90	27	62	0.56	0.76
192	5	42	0.88	0	0	N/A	1	10	0.90	0.81
194	3	4	0.25	0	5	1.00	4	11	0.64	0.83
179	0	6	1.00	3	5	0.40	0	0	N/A	0.88

XML fragments that were judged as inappropriate in XML fragment retrieval based on the proposed method did not rank in the top 1,500 of all fragments; therefore, the recall-precision curves were almost the same. Moreover, average precisions in the INEX 2004 relevance assessment were smaller than those in INEX 2003. Our XML fragment retrieval system tends to retrieve relatively small XML fragments, so it could not retrieve large XML fragments whose root node is `article`, `bdy`, or `fm`. As a result, the exhaustivity of our system tends to be small, while its specificity tends to be large and average precision becomes small. This characteristic of our XML fragment retrieval system has negative effects on average precision; therefore, we have to propose a new term weighting scheme for XML fragment retrieval. Currently, some term weighting schemes, including ours, are already published[21, 6, 10, 9, 11]; however, they are not suitable for XML fragments that overlap each other. Consequently, we will have to adopt another weighting scheme that is suitable for XML fragment retrieval[2].

In order to investigate the problems of the proposed method, we also calculated the number of indexed XML fragments by using the proposed method relative to the number of answer XML fragments in the relevance assessment.

Table 2 shows the success ratio, which is the ratio of the number of answer XML fragments indexed by the proposed method relative to the number of all answer XML fragments. In this table, we show the topic IDs whose success ratios were in the bottom five topics. Almost all success ratios of CO-topics of the INEX 2003/2004 relevance assessments were more than 90%; however, only the success ratios of these topics listed in Table 2 were remarkably low. In topic 185, especially, the higher the exhaustivity and the specificity, the smaller the success ratio. Moreover, the number of fragments that were inappropriate in our method but were answers in the relevance assessment was extremely large. In our opinion, there is a tendency that keyword-based queries (CO-topics) are used to retrieve such document fragments that just contain specified keywords, and not for searching fragments that are most suitable for the users' intention. If this concept is valid, the CO-topics whose answers contain small XML fragments

[2] Length normalization of XML fragments [18] is one of the term weighting schemes in XML fragment retrieval. We believe that not only normalization of the length of XML fragments but also the frequencies of tokens in XML fragments are important for improving the average precision of our system.

Table 3. Comparison of APD with MPD in INEX 2004 relevance assessment

	query processing (s/topic)	average precision
APD	25.48	0.0263
MPD	13.03	0.0286

should not be used for the relevance assessment in XML fragment retrieval. Consequently, we think that not only the controversial issue of the term weighting scheme for XML fragment retrieval but also inappropriate CO-topics for the INEX relevance assessment cause the low average precision of our system.

On the other hand, the number of indexed XML fragments was significantly reduced by adopting our method in the manner done for INEX 2003, so the query processing time was reduced as shown in Table 3. Therefore, XML fragment retrieval systems could perform index construction and query processing more efficiently than current systems if we adopted our method.

3.3 Discussion About the Method

Through the experiments on INEX 2003 and 2004 relevance assessments, we found that index construction time and query processing time were reduced by adopting our method based on the dynamics of terminology in quantitative linguistics. Moreover, the average precision of an XML fragment retrieval system adopting our method did not become worse. As a result, the proposed method helps to improve the performance of XML fragment retrieval systems. We are now working to improve the average precision of our XML fragment retrieval system. We expect that a novel term weighting scheme for XML fragment retrieval and a phrase match function will enable us to improve the average precision of our XML fragment retrieval system.

4 Implementing XML Fragment Retrieval System on Relational XML Database

It goes without saying that not only accuracy but also performance is an essential aspect of an XML fragment retrieval system. In fact, this is not an easy task because we have to deal with several millions of fragments extracted from document collection. In our project, we have been attempting to develop an XML fragment retrieval system based on relational databases. The reason for using relational databases is that we can utilize a variety of techniques, such as query optimization, storage management, and top-k ranking, to speed up the process of XML fragment retrieval.

This section describes our first attempt at constructing such an XML fragment retrieval system. The system is based on a path-based relational XML database system, XRel [22], that is used for storing and retrieving XML documents by using off-the-shelf relational databases. In fact, we make an extension to XRel, which originally supports XPath as its basic query language, for supporting IR queries including CO- and CAS-topics.

4.1 An Overview of XRel

The Basics. XRel [22] is a scheme to realize XML databases on top of off-the-shelf relational databases. Using XRel, we can store any well-formed (or valid) XML documents in a relational database and can retrieve XML fragments from the database by using XPath expressions.

For storing XML documents, we shred the documents into small fragments so that they can be stored in relational tuples. Actually, we take a path-based approach, in which each node in an XML tree, such as element node, attribute node, and text node, is extracted and stored in a relational table with its *simple path expression* from the root and the *region* in the document. Here, a region is represented as a pair of integers (*start*, *end*), where *start* and *end* represent the starting and ending byte positions of the node in the XML file, respectively. This information is necessary and sufficient to retain the topology of an XML tree, and we can therefore achieve lossless decomposition of XML documents into flat relational tables. An important notice here is that, for given regions, we can detect the relationships among XML nodes, such as ancestor, descendant, precedes, and follows, by applying a subsumption theorem[3] [22, 7]. Optionally, *depth*, which represents the depth of a node from the root, may be added as the third dimension in a region. In this case, we can additionally detect parent and child relations.

Schema Design. The components extracted from an XML document are stored in relational tables. Actually, there are countless ways to design the relational schema. In XRel, we decided to use four kinds of tables according to the node types, namely, *Element*, *Attribute*, *Text*, and *Path*. In addition, metadata about XML files, such as location, size, and identifier, are stored in the *Document* table. The actual schema definition of the tables are as follows:

```
Document (docID, filepath, length)
Element (docID, elementID, parentID, depth, pathID, start, end, index,
    reindex)
Attribute (docID, elementID, pathID, start, end, value)
Text (docID, elementID, pathID, start, end, value)
Path (pathID, pathexp)
```

In this definition, metadata are stored in the *Document* table with unique IDs. Also, all possible path expressions are stored in the *Path* table as character strings with their unique IDs. The other tables refer to these values in terms of `docID` and `pathID` attributes. For the *Element* table, each element node is stored with its document ID (`docID`), path expression (`pathID`), and region (`start`, `end`, and `depth`). Additionally, `elementID`, which is the unique identifier of an element node, is included for efficiency reasons, although this information is not mandatory. Likewise, `parentID`, which refers to the `elementID` of its parent node, is defined so that parent nodes can be easily reached. The `index` (`reindex`)

[3] Node x is an ancestor (descendant) of node y iff the region of x subsumes (is subsumed by, respectively) the region of y.

```
<vol no="1">
  <article>
    <title>TITLE1</title>
    <body>The first content.</body>
  </article>
  <article>
    <title>TITLE2</title>
    <body>The second <em>content.</em></body>
  </article>
</vol>
```

Fig. 4. An example XML document

attribute represents the (reverse) ordinal of nodes that share the same parent and the same path expression, and it is used to speed up positional predicates, such as `/book/author[2]` (`/book/author[-2]`). For the *Attribute* and *Text* tables, all attributes, except for `value`, act as in the *Element* table. The `value` attribute is used to store textual values of attribute and text nodes.

Figure 5 demonstrates how an XML document in Figure 4 is decomposed and stored in the relational tables.

Query Processing in XRel. For query retrieval, XRel supports XPath core, which is a subset of XPath [8], as its query language. Simply speaking, XPath core permits using "/" and "//" as location steps and using typical predicate expressions. Given an XPath core expression, XRel translates it into an equivalent SQL query that operates on the relational tables. The point here is that the translated query can be processed solely by the underlying relational database system. Then, the query result is obtained in the form of a result table, which is, in turn, reconstructed as the resultant XML fragments. For example, an XPath core query, "`//article/title[2]`," can be expressed as:

```
SELECT e1.docID, e1.st, e1.ed
FROM Path p1, Element e1
WHERE p1.pathexp LIKE '#%/article#/title'
AND e1.pathID = p1.pathID AND e1.idx = 2
ORDER BY e1.docID, e1.st
```

We do not go into the details due to the limitations of space, but more complicated queries containing node tests and/or predicates can be expressed in this way[22].

4.2 Supporting IR Queries in XRel

Statistics. Although the above tables are sufficient for processing XPath core queries, when considering INEX tasks, we need more information regarding IR statistics in order to support IR queries like in CO- and CAS-topics. To this end, we are attempting to maintain the statistics of XML nodes, in addition to the basic tables of XRel. These values include TF-IDF scores (including several variations), numbers of descendant elements, and various kinds of statistics. The concrete definition of the relational tables is as follows:

(a) Document

docID	filepath	length
0	"/path/to/foo.xml"	203

(c) Attribute

docID	elemID	pathID	st	ed	value
0	0	1	1	1	"1"

(b) Element

docID	elemID	parID	depth	pathID	st	ed	idx	reidx
0	0	-1	1	0	0	202	1	1
0	1	0	2	2	15	98	1	2
0	2	1	3	3	29	49	1	1
0	3	1	3	4	55	85	1	1
0	4	0	2	2	102	195	2	1
0	5	4	3	3	116	136	1	1
0	6	4	3	4	142	182	1	1
0	7	6	4	5	159	175	1	1

(d) Text

docID	elemID	pathID	st	ed	value
0	2	3	36	41	"TITLE1"
0	3	4	61	78	"The first content."
0	5	3	123	128	"TITLE2"
0	6	4	148	158	"The second "
0	7	5	163	170	"content."

(e) Path

pathID	pathexp
0	"#/vol"
1	"#/vol#/@no"
2	"#/vol#/article"
3	"#/vol#/article#/title"
4	"#/vol#/article#/body"
5	"#/vol#/article#/body#/em"

Fig. 5. A storage example of XRel

```
Token (docID, elementID, nodeFlag, token, articleNo, tf, tfidf,
       tfidfMG, tfief, tfipf, tfOrder)
DescendantElementNum (docID, elementID, elementName, count)
ElementStatistics (docID, elementID, sentenceNum, termFreq,
       tokenFreq, wordFreq)
```

Let us take a closer look at the definitions. The *Token* table is for storing every occurrence of a distinct token. A token is stored with the document ID, element ID, and article ID where it appears, term frequency (`tf`), and several variations of term scores (`tfidf`, `tfidfMG`, `tfief`, and `tfipf`). `tfOrder` is used for ordering the tuples in the descending order of `tf`, so as to speed up table scans. The *DescendantElementNum* table maintains the number of descendant elements for each element. The *ElementStatistics* table is for storing various kinds of statistics regarding elements, such as the numbers of elements, term frequencies, token frequencies, and word frequencies.

Processing CO-Topics. Using the above tables as well as the basic XRel tables, we can express any CO-topic in the form of `[key_1, ..., key_l, +plus_1, ..., +plus_m, -minus_1, ..., -minus_n]`, as an SQL query:

```
SELECT docID, elementID, SUM(t.tfidf) result
FROM token t
WHERE t.token IN ('key_1', ..., 'key_l')
GROUP BY docID, elementID
HAVING (SELECT COUNT(*)
        FROM token
        WHERE token IN ('minus_1', ..., 'minus_n')
        AND t.docID = token.docID
        AND t.elementID = token.elementID ) = 0
AND
       (SELECT COUNT(*)
        FROM token
        WHERE token IN ('plus_1', ..., 'plus_m')
        AND t.docID = token.docID
        AND t.elementID = token.elementID ) = m
ORDER BY result DESC;
```

As can be seen, the calculation of TF-IDF is implemented in terms of an aggregation function. It should also be noted that, in the translated query, "+key" and "-key" are expressed in terms of a HAVING clause. The resulting query is sorted in descending order of TF-IDF scores, by the ORDER BY clause.

In the same way, we can express CO-topics with phrase match by using the `value` attribute in the *Text* table. However, this may not be realistic from the viewpoint of efficiency, due to the fact that the cost for approximate matching in SQL is quite expensive. Consequently, a naive implementation would cause serious performance degradation. Actually, we may need an additional index that supports full-text search of text contents to deal with phrase matching.

4.3 Discussions About the Implementation

As discussed above, our system currently only supports XPath core and CO queries, and we thus need to accomplish further development to extend its ability and improve system performance. We are now working to improve overall system performance. In our scheme, we use a novel technique to reduce the number of result candidates. Also, we are working to achieve support of CAS- (VCAS-) topics. Efficient execution of top-k ranking in CO- and CAS-topics is another important issue.

5 Conclusion

In this paper, we proposed a method for determining XML fragments that are appropriate in keyword-based XML fragment retrieval based on the dynamics of terminology. Through some experimental evaluations in 3.1, 3.2, we found that proposed method helps to improve the performance of our XML fragment retrieval system. Moreover, we applied our XML fragment retrieval system on a relational database system, which enabled us to reduce the query processing time of our system. If we implemented a phrase match function in our system, we

could expect to improve average precision. Currently, we also have the problem of finding a term weighting scheme that is suitable for XML fragment retrieval and query optimization with a ranking function on relational database systems. These problems are the immediate tasks of our project, so we intend to solve these tasks in the near future. As an original approach, we are focusing on XML fragment retrieval without scheme information; accordingly we are going to address these problems with a view to the heterogeneous collection track of the INEX project.

Acknowledgments

This work is partly supported by the Ministry of Education, Culture, Sports, Science and Technology (MEXT), Japan, under grants #15200010, #16016243 and #16700103.

References

1. S. Agrawal, S. Chaudhuriand, and G. Das. DBXplorer: A System for Keyword-Based Search over Relational Databases. In *Proc. of the 18th International Conference on Data Engineering*, pages 5–16. IEEE CS Press, Feb./Mar. 2002.
2. S. Amer-Yahia, C. Botev, S. Buxton, P. Case, J. Doerre, D. McBeath, M. Rys, and J. Shanmugasundaram. XQuery 1.0 and XPath 2.0 Full-Text. `http://www.w3.org/TR/xmlquery-full-text/`, July 2004. W3C Working Draft 09 July 2004.
3. S. Amer-Yahia and P. Case. XQuery 1.0 and XPath 2.0 Full-Text Use Cases. `http://www.w3.org/TR/xmlquery-full-text-use-cases/`, July 2004. W3C Working Draft 09 July 2004.
4. S. Boag, D. Chamberlin, M. F. Fernandez, D. Florescu, J. Robie, and J. Siméon. XQuery 1.0: An XML Query Language. `http://www.w3.org/TR/xquery`, Oct. 2004. W3C Working Draft 29 October 2004.
5. T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau. Extensible Markup Language (XML) 1.0 (Third Edition). `http://www.w3.org/TR/REC-xml`, Feb. 2004. W3C Recommendation 04 February 2004.
6. J.-M. Bremer and M. Gertz. XQuery/IR: Integrating XML Document and Data Retrieval. In *Proc. of the 5th International Workshop on the Web and Databases (WebDB2002)*, pages 1–6. June 2002.
7. S.-Y. Chien, V. J. Tsotras, C. Zaniolo, and D. Zhang. Storing and querying multiversion XML documents using durable node numbers. In *Proc. of the 2nd International Conference on Web Information Systems Engineering*, pages 270–279. 2001.
8. J. Clark and S. DeRose. XML Path Language (XPath) Version 1.0. `http://www.w3.org/TR/xpath`, Nov. 1999. W3C Recommendation 16 November 1999.
9. S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv. XSEarch: A Semantic Search Engine for XML. In *Proc. of 29th International Conference on Very Large Data Bases*, pages 45–56. Morgan Kaufmann, Sep. 2003.

10. C.J. Crouch, S. Apte and H. Bapat. Using the Extended Vector Model for XML Retrieval. In *Proc. of the 1st Workshop of the Initiative for the Evaluation of XML Retrieval (INEX)*, pages 95–98. ERCIM, March 2003.
11. H. Cui, J.-R. Wen and T.-S. Chua. Hierarchical Indexing and Flexible Element Retrieval for Structured Document. In *Proc. of the 25th European Conference on Information Retrieval Research (ECIR2003)*. pages 73–87, April 2003.
12. N. Gövert, N. Fuhr, M. Abolhassani, and K. Großjohann. Content-Oriented XML Retrieval with HyREX. In *Proc. of the First Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 26–32. ERCIM, Mar. 2003.
13. L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram. XRANK: Ranked Keyword Search over XML Documents. In *Proc. of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 16–27. ACM Press, June 2003.
14. 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. Mar. 2004.
15. K. Hatano, H. Kinutani, M. Yoshikawa, and S. Uemura. Information Retrieval System for XML Documents. In *Proc. of the 13th International Conference on Database and Expert Systems Applications*, volume 2453 of *LNCS*, pages 758–767. Springer, Sep. 2002.
16. V. Hristidis, Y. Papakonstantinou, and A. Balmin. Keyword Proximity Search on XML Graphs. In *Proc. of the 19th International Conference on Data Engineering*, pages 367–378. IEEE CS Press, Mar. 2003.
17. K. Kageura. *The Dynamics of Terminology.* John Benjamins, 2002.
18. J. Kamps, M. de Rijke, and B. Sigurbjörnsson. Length Normalization in XML Retrieval. In *Proc. of the 27th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pages 80–87. ACM Press, July 2004.
19. M. Kaszkiel and J. Zobel. Passage Retrieval Revisited. In *Proc. of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 178–185. ACM Press, July 1997.
20. J. Nielsen. Do Websites Have Increasing Returns? `http://www.useit.com/alertbox/9704b.html`, Apr. 1997. Jakob Nielsen's Alertbox for April 15, 1997.
21. D. Shin, H. Jang and H. Jin. BUS: An Effective Indexing and Retrieval Scheme in Structured Documents In *Proc. of the 3rd ACM Conference on Digital libraries (DL'98)*, pages 235–243. June 1998.
22. M. Yoshikawa, T. Amagasa, T. Shimura, and S. Uemura. XRel: A Path-Based Approach to Storage and Retrieval of XML Documents using Relational Databases. *ACM Transactions on Internet Technology*, 1(1):110–141. June 2001.