Content-oriented XML retrieval with HyREX

Norbert Gövert
University of Dortmund
Germany

Mohammad Abolhassani
Norbert Fuhr Kai Großjohann
University of Duisburg-Essen
Germany

goevert@ls6.cs.uni-dortmund.de
{mohasani,fuhr,kai}@is.informatik.uni-duisburg.de

1 Introduction

The e*X*tensible *M*arkup *L*anguage (XML)[1] is the emerging standard for representing knowledge in almost arbitrary applications. At least almost every kind of knowledge can be represented in XML. The major purpose of XML markup is the explicit representation of the logical structure of a document. From an information retrieval (IR) point of view, users should benefit from the structural information inherent in XML documents. The *X*ML *I*nformation *R*etrieval *Q*uery *L*anguage (XIRQL) [Fuhr & Großjohann 01, Fuhr & Großjohann 02] has been developed to serve this purpose. XIRQL extends the XPath [Clark & DeRose 99] part of the (proposed standard) query language XQuery [Chamberlin et al. 01] by features important in IR style applications.

For instance, IR research has shown that document term weighting as well as query term weighting are crucial concepts for effective information retrieval. XIRQL allows for term weighting with regard to the components of the documents' logical structure. This is used for implementing the retrieval paradigm suggested by the FERMI multimedia model for IR [Chiaramella et al. 96]: Instead of treating documents as atomic units, we aim at retrieving those document *components* (elements) which answer a given information need in the *most specific* way. This strategy is used to process the *content-only (CO)* topics provided within the *IN*itiative for the *E*valuation of *X*ML retrieval (INEX)[2], where no structural conditions are used within the queries.

Given the logical structure inherent to XML documents, users want to pose queries not only on content but also on the structure of the documents. The INEX *content-and-structure (CAS)* topics reflect that. As an extension of XPath, the XIRQL query language is capable of processing these queries.

The *Hy*per-media *R*etrieval *E*ngine for *X*ML (HyREX)[3] [Abolhassani et al. 02] provides an implementation of the XIRQL query language. In the following we describe its implementation with regard to processing the INEX CO and CAS topics. In Section 2 we show how ranking of most specific document components is done in HyREX, thus serving for processing the content-only topics. Section 3 details the algorithms used to produce such a ranking of document components while Section 4 displays the evaluation results of our approach.

Section 5 shows how XIRQL concepts are used in order to process the CAS topics. In addition we give a brief overview on the concepts of data types and vague predicates which can lead to high precision searches, in combination with structural retrieval. A conclusion and an outlook on further research is given in Section 6.

2 Weighting and ranking

Classical IR models treat documents as atomic units, whereas XML suggests a tree-like view on documents. Given an information need without structural constraints, the FERMI multimedia model for IR [Chiaramella et al. 96]

[1] `http://www.w3c.org/XML/`
[2] `http://qmir.dcs.qmw.ac.uk/INEX/`
[3] `http://www.is.informatik.uni-duisburg.de/projects/hyrex/`

suggests that a system should always retrieve those document components (elements) which answer the information need in the *most specific* way.

This retrieval strategy has been implemented in HyREX in order to process the INEX content-only topics. Here we outline how classical weighting formulas (for plain document retrieval) can be generalised for structured document retrieval. Further details can be found in [Fuhr & Großjohann 01] and [Fuhr & Großjohann 02].

In analogy to the traditional plain documents, we first have to define the "atomic" units within structured documents. Such a definition serves two purposes:

- For relevance-oriented search, where no type of result element is specified, these units are the retrievable units. They provide a context within a document which can serve as a meaningful answer to a user's information need.
- Given these units, we can apply for example some kind of $\mathrm{tf} \cdot \mathrm{idf}$ formula for term weighting.

We start from the observation that text is contained in the leaf nodes of the XML tree only. These leaves would be an obvious choice as atomic units. However, this structure may be too fine-grained (e. g. markup of each item in an enumeration list, or markup of a single word in order to emphasise it). A more appropriate solution is based on the concept of *index nodes* from the FERMI multimedia model: Given a hierarchic document structure, only nodes of specific types form the roots of index nodes. In the case of XML, this means that the database administrator has to specify the names of the elements that are to be treated as index nodes.

From the weighting point of view, index nodes should be disjoint, such that each term occurrence is considered only once. On the other hand, we should allow for retrieval of results of different granularity: For very specific queries, a single paragraph may contain the right answer, whereas more general questions could be answered best by returning a whole chapter of a book. Thus, nesting of index nodes should be possible. In order to combine these two views, we first start with the most specific index nodes. For the higher-level index nodes comprising other index nodes, only the text that is not contained within the other index nodes is indexed. Using this notion of index nodes an index node tree structure is induced onto the documents. As an example, assume that we have defined *section*, *chapter*, and *book* elements as index nodes; the corresponding disjoint text units are marked as dashed boxes in the example document tree in Figure 1.

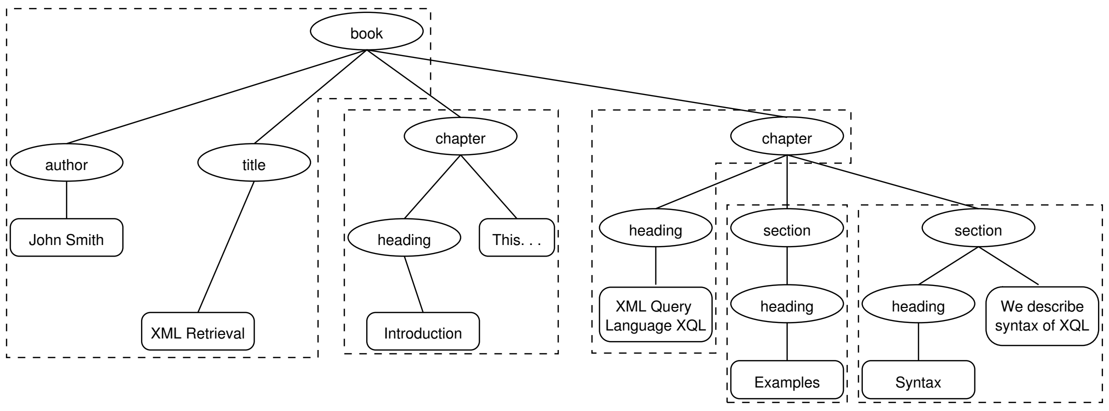

Figure 1: Example XML document tree with index nodes at the `boot`, `chapter`, and `section` levels.

Thus we have a method for computing term weights and we can do relevance-oriented search. For this, we must be able to retrieve index nodes at all levels. The indexing weights of terms within the most specific index nodes are given directly. For retrieval of the higher-level objects, we have to consider that their content is made up by the content of the index node under consideration plus the content of the descendent index nodes. Therefore, for a given index node its term weights have to be combined with the term weights of the descendant index nodes. For example, assume the following document structure, where we list the weighted terms instead of the original text:

```
<chapter> 0.3 XQL
  <section> 0.5 example </section>
  <section> 0.8 XQL 0.7 syntax </section>
</chapter>
```

A straightforward possibility would be the OR-combination of the different weights for a single term. However, searching for the term ‘XQL’ in this example would retrieve the whole chapter in the top rank, whereas the second section would be given a lower weight. It can easily be shown that this strategy always assigns the highest weight to the most general element. This result contradicts the structured document retrieval principle mentioned before. Thus, we adopt the concept of *augmentation* from [Fuhr et al. 98]. For this purpose, index term weights are down weighted (multiplied by an augmentation factor) when they are propagated upwards to the next index node. In our example, using an augmentation factor of 0.6, the retrieval weight of the chapter w. r. t. to the query ‘XQL’ would be $0.3 + 0.6 \cdot 0.8 - 0.3 \cdot 0.6 \cdot 0.8 = 0.596$, thus ranking the section ahead of the chapter.

3 Retrieval algorithm

For doing relevance-oriented searches, the XIRQL query language defines the respective relevance selection operator ‘`inode()`’ and the relevance projection operator ‘`...`’. However, in our INEX experiments we bypassed the XIRQL logical layer and directly accessed HyREX’s physical layer in order to develop an efficient retrieval strategy for processing the INEX content-only topics.

The parallel algorithm which is described in the following, uses direct access to the inverted lists of the query terms in a given topic. As a prerequisite for the algorithm it is assumed that the inverted lists contain all the details necessary to describe a term occurrence for our index node retrieval approach:

Index node identifier: Each index node is assigned an ID during indexing.

Index node description: An index node is described by a path, beginning from the document root to the index node itself. The path contains the index node identifiers of all the index nodes of which borders are crossed, together with their respective augmentation factor.

Weight: This is the indexing weight for the given term within the index node represented.

Furthermore it is assumed that the entries in the inverted lists are ordered by document identifiers on the first level, and preordering of the index nodes (as they appear in the documents) on the second level.

Given that, the algorithm processes the occurrence descriptions within the various inverted lists until all of them are read. Due to the ordering in which the occurrence descriptions are read from the inverted lists, we reach that retrieval status value (RSV) computation for a given index node can be finished as early as possible. The `read_term` method observes the inverted lists beginning at their head and delivers the occurrence description from all of the inverted lists which is next according to the ordering scheme described above:

readterm() : inode_id, inode_path, augmentation, term, weight
Method that implements a priority queue for the candidate set of occurrence descriptions to be processed next; these are read directly from the inverted lists of the query terms.

inode_path[l] Array variable that lists the index node ids which make up the path from the document root towards the index node considered.

augmentation[l] Array variable that lists the augmentation factors belonging to the index nodes represented by the `inode_path` array.

term Identifier of the inverted list from which the current occurrence description is read.

weight Term weight within the index node referenced by `inode_id`.

Within the outer loop of the algorithm occurrence descriptions for all of the query terms are read until all the respective inverted lists are processed:

```
while (inode_id, inode_path, augmentation, term, weight) = readterm() do
    level = length inode_path
    ...
od
```

Figure 2 displays the inner part of the loop. First, it is checked whether there are index nodes, for which all information for computation of the RSV is available. Where this is the case, the RSV is computed and the index node is pushed into the set of result candidates for the ranking. The following variables are needed for this:

qterm_weights[t] Array variable which lists the query term weights.

cumulated_weights[l, t] Matrix variable for cumulated index weights for t query terms at l index node levels.

lastlevel Level of the index node which has been processed in the previous iteration of the `while` loop.

lastnodes[lastlevel] Array variable representing the path of index nodes leading to the index node which has been processed in the previous iteration of the `while` loop.

add_result(inode_id, weight) Method to add an index node together with its respective RSV to the set of result candidates.

Before applying the retrieval function to an index node the contribution of the descendent index objects within the path represented by `lastnodes` to the term weights needs to be computed. The term weights are propagated beginning from the leaf in `lastnodes`; at each index node border they are reduced by means of an augmentation factor given for the specific index object. After an index object is processed this way the respective term weights in the `cumulated_weights` matrix is reset.

When the RSVs for the index nodes finished have been processed this way, the `lastnodes` vector is set to the path to the current index object under consideration. The weight of the term under consideration is stored within the `cumulated_weights` matrix.

```
for j = 0 to min(level, lastlevel) do
    // check if some index nodes are finished
    if lastnodes[j] <> inodes[j] then
        // compute RSVs for finished index nodes
        for i = lastlevel downto j do
            // apply linear retrieval function (scalar value)
            rsv = cumulated_weights[i] * qterm_weights
            add_result(lastnode[i], rsv)
            // propagate term weights towards the root
            if i > 1 then
              cumulated_weights[i - 1] = cumulated_weights[i - 1]
                | augmentation[i] & cumulated_weights[i]
            fi
            // reset cumulated weights
            cumulated_weights[i] = 0
        od
        last // exit loop
    fi
od
lastnodes = inodes
lastlevel = level
// store weight of occurrence for current term
cumulated_weights[level, term] = weight
```

Figure 2: Parallel algorithm for processing content-only topics

After all occurrence descriptions are processed, the result can be delivered to the user. If there is a maximum number n of result items to be retrieved (for INEX this was 100), the `add_result` method can use a heap structure for selecting the n top ranking elements from the set of all index nodes processed.

The algorithm described here is efficient in terms of memory usage. By processing the inverted lists in parallel we achieve that retrieval status values for an index node once touched can be computed as early as possible. It follows that the number of accumulators for intermediary results is bounded by the maximum level an index node can have.

An alternative algorithm which processes the inverted lists sequentially would not be able to compute the final retrieval status values until all inverted lists are read. Thus it would have to allocate accumulators for all index nodes ever touched within the inverted lists of the query terms.

4 Evaluation of effectiveness

One of the results of the first INEX workshop 2002 has been the definition of a metrics for evaluation of the effectiveness of content-oriented XML retrieval approaches [Gövert & Kazai 03]. This metrics, based on the notion of recall and precision, has been used here for evaluation, together with the relevance assessments package version 1.7 (available from the INEX {down,up}load area[4]).

Our focus has been on experimenting with different augmentation factors when doing the relevance-oriented retrieval described in Section 2. Figure 3 show the recall / precision curves for six different augmentation factors from 0.0 to 1.0, step 0.2. For each plot the top 100 results from the rankings have been accounted for. From the graphs one can see that small augmentation factors in the range from 0.2 to 0.4 should be used for most effective content-oriented XML retrieval.

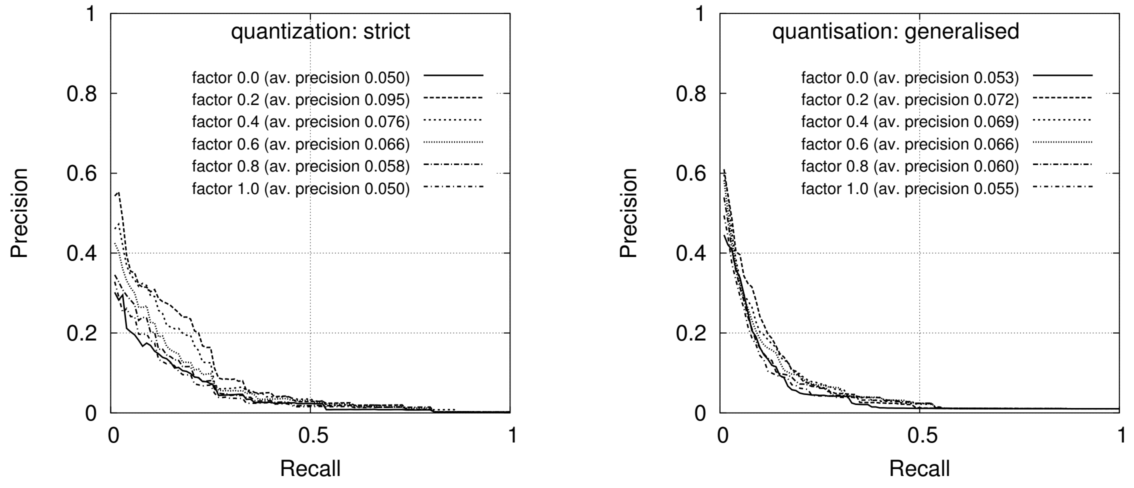

Figure 3: Recall / precision curves for different augmentation factors (content-only topics).

5 XIRQL: Processing content-and-structure topics

The XIRQL query language can be used to query on structured document collections using content *and* structural conditions. Given a fine-grained markup of XML documents, a mapping of the elements to specific data types (e. g. person names, dates, technical measurement values, names of geographic regions) can be done. For these data types special search predicates are provided, most of which are vague (e. g. phonetic similarity of names, approximate matching of dates, closeness of geographic locations). The concept of data types and vague search predicates [Fuhr 99] can thus be used to enhance the precision of a given information need.

These features have been used to process the INEX content-and-structure topics. For this, the CAS topics have been converted to XIRQL in a fully automatic way and then have been processed with HyREX. As an example, topic 24 is displayed in Figure 4. Figure 5 shows the result of its conversion into XIRQL syntax. The topic is about retrieval of articles, thus the respective XPath expression `/article` starts the query. The further constraints are specified by filters which combine various conditions via weighted sum operators. The set of conditions in the first weighted sum results from the structural conditions within the title section of the original topic. For different elements specific search predicates are applied (phonetic similarity on author names and stemmed search for other query terms). The second set of conditions results from the query terms in the description and keywords section of

[4] `http://ls6-www.cs.uni-dortmund.de/ir/projects/inex/download/`

```
<INEX-Topic topic-id="24" query-type="CAS">
  <Title>
    <te>article</te>
    <cw>Smith Jones</cw>                                  <ce>au</ce>
    <cw>software engineering and process improvement</cw> <ce>bdy</ce>
  </Title>
  <Description>
    Find articles about software process improvement by the programming industry
    that are written by an author we believe is named either Smith or Jones.
  </Description>
  <Narrative>
    Only documents about software engineering written by Capers Jones are relevant.
  </Narrative>
  <Keywords>
    Smith Jones software engineering and  process improvement programming
  </Keywords>
</INEX-Topic>
```

Figure 4: CAS topic 24 in XML format

topic 24. We use relevance-oriented search for them, so that documents where all terms are in the same index node are boosted. The figures in front of the various conditions denote the (non-normalised) query term weights (the weighted sum operator normalises these weights internally). Some CAS topics include phrases which are emulated by requiring all terms to be in the same text node. For example, one component of the weighted sum could be as follows:

```
.//au//#PCDATA[. $soundex$ "John" $and$ . $soundex$ "Smith"]
```

The "`//#PCDATA`" part in the structural conditions is required for implementation-related reasons.

6 Conclusion

We have shown how HyREX has been utilised to process the INEX tasks. For dealing with the content-only topics an algorithm based on the notion of index nodes and augmentation of index term weights has been developed. The XIRQL query language has been used to process the content-and-structure topics.

A first evaluation could show how index term weights can be augmented for effective content-oriented XML retrieval. For further improvements alternative approaches for selecting appropriate augmentation factors are to be tested. In principle, augmentation factors may need to be different for each index node. A good compromise between these specific weights and a single global weight may be the definition of type-specific weights, i. e. depending on the name of the index node root element. The optimum choice between these possibilities will be subject to theoretical and empirical investigations. Another way to derive augmentation factors could be based on information about the size of index nodes and the number of siblings and children. Finally, having relevance assessments for structured document retrieval now, one could even think of relevance feedback methods for estimating the augmentation factors. Further research will go into that direction.

Another issue is efficiency. In this article we describe an algorithm that uses all information from the inverted lists in order to compute RSVs. In order to become more efficient one can think of variants which terminate earlier. Here, the trade-off between efficiency and effectiveness has to be considered.

References

Abolhassani, M.; Fuhr, N.; Gövert, N.; Großjohann, K. (2002). *HyREX: Hypermedia Retrieval Engine for XML*. Research report, University of Dortmund, Department of Computer Science, Dortmund, Germany.

Chamberlin, D.; Florescu, D.; Robie, J.; Siméon, J.; Stefanescu, M. (2001). *XQuery: A Query Language for XML*. Technical report, World Wide Web Consortium.

```
/article[
  wsum(
    1, .//au//#PCDATA  $soundex$ "Jones",
    1, .//au//#PCDATA  $soundex$ "Smith",
    1, .//bdy//#PCDATA $stemen$  "engineering",
    1, .//bdy//#PCDATA $stemen$  "improvement",
    1, .//bdy//#PCDATA $stemen$  "process",
    1, .//bdy//#PCDATA $stemen$  "software"
  )]//*[wsum(
    1, ...             $stemen$  "Find",
    2, ...             $stemen$  "Jones",
    2, ...             $stemen$  "Smith",
    1, ...             $stemen$  "articles",
    1, ...             $stemen$  "author",
    1, ...             $stemen$  "believe",
    1, ...             $stemen$  "engineering",
    2, ...             $stemen$  "improvement",
    1, ...             $stemen$  "industry",
    1, ...             $stemen$  "named",
    2, ...             $stemen$  "process",
    2, ...             $stemen$  "programming",
    2, ...             $stemen$  "software",
    1, ...             $stemen$  "written" ) ]
```

Figure 5: CAS topic 24 in XIRQL syntax

Chiaramella, Y.; Mulhem, P.; Fourel, F. (1996). *A Model for Multimedia Information Retrieval*. Technical report, FERMI ESPRIT BRA 8134, University of Glasgow.

Clark, J.; DeRose, S. (1999). *XML Path Language (XPath) Version 1.0*. Technical report, World Wide Web Consortium.

Fuhr, N.; Großjohann, K. (2001). XIRQL: A Query Language for Information Retrieval in XML Documents. In: Croft, W.; Harper, D.; Kraft, D.; Zobel, J. (eds.): *Proceedings of the 24th Annual International Conference on Research and development in Information Retrieval*, pages 172–180. ACM, New York.

Fuhr, N.; Großjohann, K. (2002). *XIRQL: An XML Query Language Based on Information Retrieval Concepts*. Submitted.

Fuhr, N. (1999). Towards Data Abstraction in Networked Information Retrieval Systems. *Information Processing and Management 35(2)*, pages 101–119.

Fuhr, N.; Gövert, N.; Rölleke, T. (1998). DOLORES: A System for Logic-Based Retrieval of Multimedia Objects. In: Croft, W. B.; Moffat, A.; van Rijsbergen, C. J.; Wilkinson, R.; Zobel, J. (eds.): *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 257–265. ACM, New York.

Gövert, N.; Kazai, G. (2003). Overview of the Initiative for the Evaluation of XML retrieval (INEX) 2002. In: Fuhr, N.; Gövert, N.; Kazai, G.; Lalmas, M. (eds.): *Initiative for the Evaluation of XML Retrieval (INEX). Proceedings of the First INEX Workshop. Dagstuhl, Germany, December 8-11, 2002*, ERCIM Workshop Proceedings. ERCIM, Sophia Antipolis, France.