A Query Language and User Interface for XML Information Retrieval

Norbert Fuhr[1], Kai Großjohann[2], and Sascha Kriewel[3]

[1] University of Duisburg-Essen fuhr@uni-duisburg.de
[2] University of Duisburg-Essen kai.grossjohann@uni-duisburg.de
[3] University of Duisburg-Essen kriewel@is.informatik.uni-duisburg.de

1 Introduction

As XML is about to become the standard format for structured documents, there is an increasing need for appropriate information retrieval (IR) methods. Since classical IR methods were developed for unstructured documents only, the logical markup of XML documents poses new challenges.

Since XML supports logical markup of texts both at the macro level (structuring markup for chapter, section, paragraph and so on) and the micro level (e.g., MathML for mathematical formulas, CML for chemical formulas), retrieval methods dealing with both kinds of markup should be developed. At the macro level, fulltext retrieval should allow for selection of appropriate parts of a document in response to a query, such as by returning a section or a paragraph instead of the complete document. At the micro level, specific similarity operators for different types of text or data should be provided (such as similarity of chemical structures, phonetic similarity for person names).

Although a large number of query languages for XML have been proposed in recent years, none of them fully addresses the IR issues related to XML; especially, the core XQuery proposal of the W3C working group [4] offers no support for IR-oriented querying of XML sources; the discussion about extensions for text retrieval has started only recently (see the requirements document by Buxton and Rys [5] and the use cases by Amer-Yahia and Case [2]). There are only a few approaches that provide partial solutions to the IR problem, namely by taking into account the intrinsic imprecision and vagueness of IR; however, none of them are based on a consistent model of uncertainty (see section 5).

In this paper, we present the query language XIRQL which combines the major concepts of XML querying with those from IR. XIRQL is based on

XPath
XIRQL
XIRQL
vagueness
vagueness

XPath, which we extend by IR concepts. We also provide a consistent model for dealing with uncertainty.

For building a complete IR system, the query language and the model are not enough. One also needs to deal with user interface issues. On the input side, the question of query formulation arises: the query language allows for combinding structural conditions with content conditions, and the user interface needs to reflect this. On the output side, we observe two kinds of relationships between retrieval results. In traditional document retrieval, the retrievable items (i.e., documents) are considered to be independent from each other. This means that the system only needs to visualize the ordering imposed by the ranking. But in the case of retrieval from XML documents, two retrieved items may have a structural relationship, if they come from the same document: One could be the ancestor of another, or a sibling, and so on.

So in addition to the query language XIRQL, we describe graphical user interfaces for interactive query formulation as well as for result presentation.

This paper is structured as follows. In the following section, we discuss the problem of IR on XML documents (section 2). Then we present the major concepts of our new query language XIRQL (section 3). Our graphical user interfaces are described in section 4. A survey on related work is given in section 5, followed by the conclusions and the outlook.

2 Requirements for an XML IR Query Language

From an IR point of view, the combination of content with logical markup in XML offers the following opportunities for enhancing IR functionality in comparison to plain text:

- Queries referring to content only should retrieve relevant document parts according to the logical structure, thus overcoming the limitations of passage retrieval. The FERMI model by Chiaramella et al. [7] suggests the following strategy for the retrieval of structured (multimedia) documents: A system should always retrieve the most specific part of a document answering the query.
- Based on the markup of specific elements, high-precision searches can be performed that look for content occurring in specific elements (e.g., distinguishing between the sender and the addressee of a letter, finding the definition of a concept in a mathematics textbook). On the other hand, the intrinsic uncertainty and vagueness of IR should also be considered when interpreting structural conditions; thus, a vague interpretation of this type of conditions should be supported.
- The concept of *mixed content* allows for the combination of high precision searches with plain text search. An element contains mixed content if both subelements and plain text (#PCDATA) may occur in it. Thus, it is possible to mark up specific items occurring in a text. For example,

in an arts encyclopedia, names of artists, places they worked, and titles of pieces of art may be marked up (thus allowing for example, to search for Picasso's paintings of toreadors, avoiding passages mentioning Picasso's frequent visits to bull-fights).

XPath
XPath
XPath
vagueness
XPath

With respect to these requirements, XPath seems to be a good starting point for IR on XML documents. However, the following features should be added to XPath:

Weighting. IR research has shown that document term weighting as well as query term weighting are necessary tools for effective retrieval in textual documents. So comparisons in XPath referring to the text of elements should consider index term weights. Furthermore, query term weighting should also be possible, by introducing a weighted sum operator (allowing conditions like $0.6 \cdot$ "XML" $+ 0.4 \cdot$ "retrieval"). These weights should be used for computing an overall retrieval status value for the elements retrieved, thus resulting in a ranked list of elements.

Relevance-oriented search. The query language should also support traditional IR queries, where only the requested content is specified, but not the type of elements to be retrieved. In this case, the IR system should be able to retrieve the most relevant elements; following the FERMI multimedia model cited above, this should be the most specific element(s) that fulfill(s) the query. In the presence of weighted index terms, the tradeoff between these weights and the specifity of an answer has to be considered, possibly by an appropriate weighting scheme.

Data types and vague predicates. The standard IR approach for weighting supports vague searches on plain text only. XML allows for fine grained markup of elements, and thus, there should be the possibility to use special search predicates for different types of elements. For example, for an element containing person names, similarity search for proper names should be offered; in technical documents, elements containing measurement values should be searchable by means of the comparison predicates $>$ and $<$ operating on floating point numbers. Thus, there should be the possibility to have elements of different data types, where each data type comes with a set of specific search predicates. In order to support the intrinsic vagueness of IR, most of these predicates should be vague (search for measurements that were taken at about 20 °C, for instance).

Structural vagueness. XPath is closely tied to the XML syntax, but it is possible to use syntactically different XML variants to express the same meaning. For example, a particular information could be encoded as an XML attribute or as an XML element. As another example, a user may wish to search for a value of a specific data type in a document (a person name, say), without bothering about the element names. Thus, appropriate generalizations should be included in the query language.

XIRQL
XPath
weighting
probabilistic event
index node

3 XIRQL Concepts

In this section, we describe concepts for integrating the features listed in the previous section in XIRQL. These are: weighting, relevance-oriented search, data types and vague predicates, and structural vagueness.

3.1 Weighting

At first glance, extending XPath by a weighting mechanism seems to be a straightforward approach. Assuming probabilistic independence, the combination of weights according to the different Boolean operators is obvious, thus leading to an overall weight for any answer. However, there are two major problems that have to be solved first: 1) How should terms in structured documents be weighted? 2) What are the probabilistic events, i.e., which term occurrences are identical, and which are independent? Obviously, the answer to the second question depends partly on the answer to the first one.

As we said before, classical IR models have treated documents as atomic units, whereas XML suggests a tree-like view of documents. One possibility for term weighting in structured documents would be the development of a completely new weighting mechanism. Given the long experience with weighting formulas for unstructured documents, such an approach would probably take a big effort to obtain good retrieval quality. As an alternative, we suggest to generalize the classical weighting formulas. Thus, we have to define the "atomic" units in XML documents that are to be treated like atomic documents. The benefit of such a definition is twofold:

1. Given these units, we can apply some kind of $\mathrm{tf} \cdot \mathrm{idf}$ formula for term weighting.
2. For relevance-oriented search, where no type of result element is specified, only these units can be returned as answers, whereas other elements are not considered as meaningful results.

We start from the observation that text is contained in the leaf nodes of the XML tree only. So these leaves would be an obvious choice as atomic units. However, this structure may be too fine-grained. (It could be the markup of each item in an enumeration list, or markup of a single word in order to emphasize 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 objects. In the case of XML, this means that we have to specify the names of the elements that are to be treated as index nodes. This definition can be part of the XML Schema (see below).

From the weighting point of view, index objects 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 objects should be possible. In order to combine these two views, we first start with the most specific index nodes. For the higher-level index objects comprising other index objects, only the text that is not contained within the other index objects is indexed. As an example, assume that we have defined section, chapter and book elements as index nodes in our example document; the corresponding disjoint text units are marked as dashed boxes in figure 1.

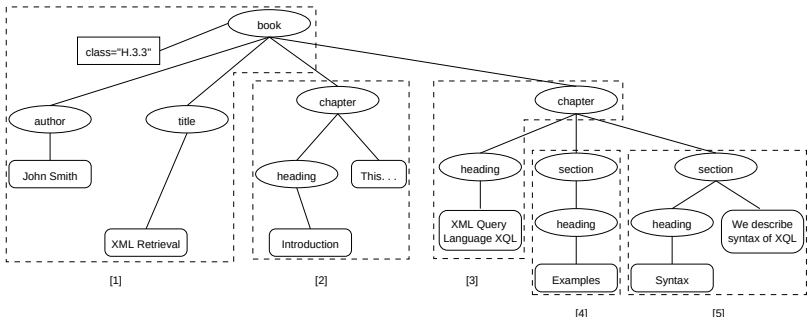

Fig. 1. Example XML document tree. Dashed boxes indicate index nodes; bracketed numbers serve as identifiers.

So we have a method for computing term weights, and we can do relevance based search. Now we have to solve the problem of combining weights and structural conditions. For the following examples, let us assume that there is a comparison predicate `cw` (contains word) which tests for word occurrence in an element. Now consider the query `//section[heading cw "syntax"]` and assume that this word does not only occur in the heading, but also multiple times within the same index node (i.e., section). Here we first have to decide about the interpretation of such a query: Is it a content-related condition, or does the user search for the occurrence of a specific string? In the latter case, in would be reasonable to view the filter part as a Boolean condition, for which only binary weights are possible. We offer this possibility by providing data types with a variety of predicates, where some of them are Boolean and others are vague (see below).

In the content-related interpretation, we use the weight from the corresponding index node. The major justification for this strategy is the fact that the meaning of a term depends heavily on its context, and this context should never be ignored in content-oriented searches — even when structural conditions are specified. These conditions should only work as additional filters. So we take the term weight from the index node. Thus the index node determines the significance of a term in the context given by the node.

event key
event key
event expression
event expression
event expression
relevance-oriented search

With the term weights defined this way, we have also solved the problem of independence/identity of probabilistic events: Each term in each index node represents a unique probabilistic event, and all occurrences of a term within the same node refer to the same event. (Both occurrences of the word "syntax" in the last section of our example document represent the same event, for example.) Assuming unique node IDs, events can be identified by event keys that are pairs [node ID, term]. Given the node IDs shown in square brackets in figure 1, the occurrence of the word "syntax" in the last section is represented by the event $[5, \text{syntax}]$. For retrieval, we assume that different events are independent. That is, different terms are independent of each other. Moreover, occurrences of the same term in different index nodes are also independent of each other. Following this idea, retrieval results correspond to Boolean combinations of probabilistic events which we call event expressions. For example, a search for sections dealing with the syntax of XQL could be specified as `//section[.//* cw "XQL" and .//* cw "syntax"]`. Here, our example document would yield the conjunction $[5, \text{XQL}] \wedge [5, \text{syntax}]$. In contrast, a query searching for this content in complete documents would have to consider the occurrence of the term "XQL" in two different index nodes, thus leading to the Boolean expression $([3, \text{XQL}] \vee [5, \text{XQL}]) \wedge [5, \text{syntax}]$.

For dealing with these Boolean expressions, we adopt the idea of event keys and event expressions described by Fuhr and Rölleke [16]. With the method described there, we can compute the correct probability for any combination of independent events (see also Fuhr and Großjohann [15]). Furthermore, the method can be extended to allow for query term weighting. Assume that the query for sections about XQL syntax would be reformulated as `//section[0.6 · .//* cw "XQL" + 0.4 · .//* cw "syntax"]`. For each of the conditions combined by the weighted sum operator, we introduce an additional event with a probability as specified in the query (the sum of these probabilities must not exceed 1). Let us assume that we identify these events as pairs of an ID referring to the weighted sum expression, and the corresponding term. Furthermore, the operator '$\cdot$' is mapped onto the logical conjunction, and '$+$' onto disjunction. For the last section of our example document, this would result in the event expression $[q_1, \text{XQL}] \wedge [5, \text{XQL}] \vee [q_1, \text{syntax}] \wedge [5, \text{syntax}]$. Assuming that different query conditions belonging to the same weighted sum expression are disjoint events, this event expression is mapped onto the scalar product of query and document term weights: $P([q_1, \text{XQL}]) \cdot P([5, \text{XQL}]) + P([q_1, \text{syntax}]) \cdot P([5, \text{syntax}])$.

3.2 Relevance-oriented Search

Above, we have described a method for combining weights and structural conditions. In contrast, relevance-based search omits any structural conditions; instead, we must be able to retrieve index objects at all levels. The index weights of the most specific index nodes are given directly. For retrieval of the higher-level objects, we have to combine the weights of the different text

units contained therein. 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 be easily 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. [14]. For this purpose, index term weights are downweighted (multiplied by an augmentation weight) when they are propagated upwards to the next index object. In our example, using an augmentation weight of 0.6, the retrieval weight of the chapter with respect 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.

For similar reasons as above, we use event keys and expressions in order to implement a consistent weighting process (so that equivalent query expressions result in the same weights for any given document). Fuhr et al. [14] introduce augmentation weights (i.e., probabilistic events) by means of probabilistic rules. In our case, we can attach them to the root element of index nodes. Denoting these events as index node number, the last retrieval example would result in the event expression $[1, \text{XQL}] \vee [3] \wedge [3, \text{XQL}]$.

In the following, paths leading to index nodes are denoted by 'inode()' and recursive search with downweighting is indicated via '...'. As an example, the query `/document//inode()[... cw "XQL" and ... cw "syntax"]` searches for index nodes about "XQL" and "syntax", thus resulting in the event expression $([1, \text{XQL}] \vee [3] \wedge [3, \text{XQL}]) \wedge [2] \wedge [2, \text{syntax}]$.

In principle, augmentation weights may 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 betweeen these possibilities will be subject to empirical investigations.

augmentation
downweighted
augmentation
augmentation
augmentation
event expression
downweighting
event expression
augmentation

3.3 Data Types and Vague Predicates

Given the possibility of fine-grained markup in XML documents, we would like to exploit this information in order to perform more specific searches. For the content of certain elements, structural conditions are not sufficient, since the standard text search methods are inappropriate. For example, in an arts encyclopedia, it would be possible to mark artist's names, locations or dates. Given this markup, one could imagine a query like "Give me information

vague predicate
vagueness
XIRQL
vague predicate
vague predicate

about an artist whose name is similar to Ulbrich and who worked around 1900 near Frankfort, Germany", which should also retrieve an article mentioning Ernst Olbrich's work in Darmstadt, Germany, in 1899. Thus, we need *vague predicates* for different kinds of data types (person names, locations, dates, and so on). Besides similarity (vague equality), additional data type-specific comparison operators should be provided (e.g., 'near', <, >, or 'broader', 'narrower' and 'related' for terms from a classification or thesaurus). In order to deal with vagueness, these predicates should return a weight as a result of the comparison between the query value and the value found in the document.

The XML standard itself only distinguishes between three data types, namely text, integer and date. The XML Schema recommendation [12] extends these types towards atomic types and constructors (tuple, set) that are typical for database systems.

For the document-oriented view, this notion of data types is of limited use. This is due to the fact that most of the data types relevant for IR applications can hardly be specified at the syntactic level (consider for instance names of geographic locations, or English vs. French text). In the context of XIRQL, data types are characterized by their sets of vague predicates (such as phonetic similarity of names, English vs. French stemming). Thus, for supporting IR in XML documents, there should be a core set of appropriate data types, and the system should be designed in an extensible way so that application-specific data types can be added easily.

We do not discuss implementation issues here, but it is clear that the system needs to provide appropriate index structures, for structural conditions and also for the (possibly vague) search predicates — both for the core and the application-specific data types, of course. This problem is rather challenging, as we suspect that separate index structures for the tree structure and for the search predicates will not be sufficient; rather, they have to be combined in some way.

Candidates for the core set are texts in different languages, hierarchical classification schemes, thesauri and person names. In order to perform text searches, some knowledge about the kind of text is necessary. Truncation and adjacency operators available in many IR systems are suitable for western languages only (whereas XML in combination with unicode allows for coding of most written languages). Therefore, language-specific predicates, e.g., for dealing with stemming, noun phrases and composite words should be provided. Since documents may contain elements in multiple languages, the language problem should be handled at the data type level. Classification schemes and thesauri are very popular now in many digital library applications; thus, the relationships from these schemes should be supported, perhaps by including narrower or related terms in the search. Vague predicates for this data type should allow for automatic inclusion of terms that are similar according to the classification scheme. Person names often pose problems in document search, as the first and middle names may sometimes be initials only (therefore, searching for "Jack Smith" should also retrieve "J. Smith",

with a reduced weight). A major problem is the correct spelling of names, especially when transliteration is involved (e.g., "Chebychef"); thus, phonetic similarity or spelling-tolerant search should be provided.

data types in IR
XIRQL
XIRQL
XIRQL

Application-specific data types should support vague versions of the predicates that are common in this area. For example, in technical texts, measurement values often play an important role; thus, dealing with the different units, the linear ordering involved ($<$) as well as similarity (vague equality) should be supported (show me all measurements taken at room temperature). For texts describing chemical elements and compounds, it should be possible to search for elements of compounds, or to search for common generalizations (search for "aluminum salts", without the need to enumerate them).

As a framework for dealing with these problems, we adopt the concept of data types in IR from Fuhr [13], where a data type T is a pair consisting of a domain $|T|$ and a set of (vague comparison) predicates $P_T = \{c_1, \ldots, c_n\}$. Like in other type systems, IR data types should also be organized in a type hierarchy (e.g., Text – Western-Language – English), where the subtype restricts the domain and/or provides additional predicates. (In the example, it could be n-gram matching for general text, plus adjacency and truncation for western languages, plus stemming and noun phrase search for English.) Through this mechanism, additional data types can be defined easily by refining the appropriate data type (introducing French as refinement of Western-Language, say)[4].

In order to exploit these data types in retrieval, the data types of the XML elements have to be defined. Although the XML Schema recommendation [12] is targeted towards the data-centric view of XML, it can also be used for our purpose. Most of the data types discussed above are simple types in terms of XML Schema (that is, they have no internal structure), but do not belong to the builtin types of XML Schema. Thus, they have to be derived by means of restriction from the builtin types. However, in most cases, it is not possible to give necessary conditions for the restriction (consider English as a specialization of normalizedString). In addition, XML Schema does not deal with (vague) predicates of data types; they can be listed as application info only and are treated like comments by the schema processor.

In principle, XIRQL queries can also be processed for collections where no DTD or XML schema is given. However, in this case, XIRQL would assume that the content of all elements belongs to the same basic data type text, for which only basic predicates (like string equality and substring search) are provided. The more information about the data type of elements is provided, the more appropriate search predicates for the different elements can be provided by XIRQL. On the other hand, when no DTD is given, it will be very difficult

[4] Please note that we make no additional assumptions about the internal structure of the text data type (and its subtypes), like representing text as set or list of words.

vagueness
XPath
XIRQL
ontologys!over element names
graphical user interface
query formulation
result presentation
XIRQL
XIRQL
XIRQL
XIRQL

for a user to formulate meaningful queries with structural conditions — most queries of this type would retrieve no documents at all.

3.4 Structural Vagueness

Since typical queries in IR are vague, the query language should also support vagueness in different forms. Besides relevance-based search as described above, relativism with respect to elements and attributes seems to be an important feature. The XPath distinction between attributes and elements may not be relevant for many users. In XIRQL, `author` searches an element, `@author` retrieves an attribute and `=author` is used for abstracting from this distinction.

Another possible form of relativism is induced by the introduction of data types. For example, we may want to search for persons in documents, without specifying their role (author, editor, referenced author, subject of a biography) in these documents. Thus, we provide a mechanism for searching for certain data types, regardless of their position in the XML document tree. For example, `#persname` searches for all elements and attributes of the data type persname.

Currently, we are working on further generalizations of structural conditions. One direction is based on ontologies over element names. For example, assuming that `region` is a subproperty of the more general element named `geographic-area`, which in turn has additional subproperties `continent` and `country`, we would expand the original element name `region` into the disjunction `region | country | continent`. The sequence of elements in a path can also be subject to vague interpretations (e.g., `author = "Smith"` should also match vaguely `author/name` and `author/name/lastname`).

4 Graphical User Interfaces for XML IR

A graphical user interface for retrieval should support two tasks, namely query formulation and result presentation. In the following, we first describe a mechanism for visually constructing XIRQL queries without having to know the syntax of the query language, and without being intimately acquainted with the document structure (DTD). Then we present two approaches for displaying the result list along with the size of answer elements and structural relationships between them.

4.1 Visual Query Formulation

With all the features of XIRQL described above, it is obvious that query formulation in XIRQL is a rather complex task. However, we assume that most end users will not have to formulate XIRQL queries, since there will be

applications which generate XIRQL queries from user input to easy-to-use interfaces. For the remaining cases where XIRQL queries have to be formulated interactively, we want to provide an appropriate user interface. This interface should support the formulation of syntactically and semantically correct queries:

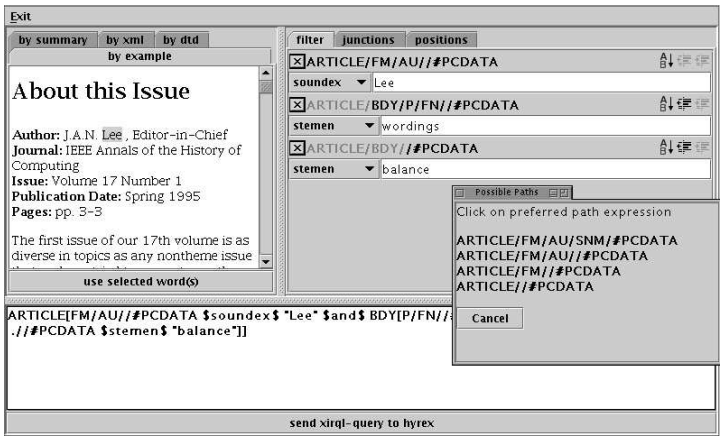

Fig. 2. Interface for formulating queries.

- With regard to query syntax, we take a menu-based approach, which covers only a subset of the full query syntax. In addition, the user can edit directly the generated XIRQL formulation.
- Semantically correct queries are supported by giving information about the structure of the XML documents to be queried, the data types of fields and the applicable predicates.

A screenshot of our interface can be seen in figure 2. There are three areas: On the left, the *structure condition area* enables users to formulate single query conditions. On the right, the *condition list area* allows users to edit the query conditions and to specify how to combine them to form the whole query. At all times, a paraphrase of the current query in XIRQL syntax is kept up to date in the *paraphrase area* at the bottom.

For formulating a single query condition, the main mechanism is *Query by Example*. It comes in three variants. In the screenshot, the layout-oriented variant is shown. The user can click on a word in that document and the system derives from it a *structural condition* (candidate) and a *value condition* (candidate). The structural condition describes the list of element names on the path from the root node to the leaf node in the XML tree. From it, a number of generalizations (using the `//` operator and the `*` wild card) are produced and shown to the user (see the popup window in the lower right of the screenshot). After selecting the structural condition, the query condition

XIRQL
XIRQL
XIRQL
structure condition area
graphical user interface!structure condition area
condition list area
graphical user interface!condition list area
XIRQL
paraphrase area
graphical user interface!paraphrase area
Query by Example
structural condition
value condition

condition list area
structural dependence
XIRQL
XIRQL
XIRQL
result presentation
result presentation!structural relationships
result presentation!size of answer elements

is added to the condition list area, where additional changes can be made: The comparison value (defaulting to the word the user selected) can be edited, and a search predicate can be chosen for this condition.

In addition to the layout-oriented variant of Query by Example, we offer a structure-oriented variant where people see an expandable tree of the XML document, as well as a structure-oriented variant which shows a document surrogate only. Finally, as an alternative to Query by Example, we offer a *DTD oriented* method for specifying the structure condition which does not rely on an example document.

The next step is to specify how the query conditions thus collected should be combined to form the whole query. Here, we focus on the *structural dependence* between the conditions. This is achieved by specifying a common prefix for two query conditions. For example, in the third condition, `/ARTICLE/BDY` is grayed out. This means the match for the second and third conditions must be in the same `BDY` element (and hence within the same `ARTICLE` element). The graying-out connects two adjacent conditions; by making it possible to move conditions up and down in the list, structual dependence between any two conditions can be expressed. In addition to the structural dependence, the *Boolean connectors* between the conditions also need to be specified. We do this in a simple manner, allowing the user to choose between `and` and `or` between any two conditions, but we plan more elaborate support, possibly based on Venn diagrams.

To test the usefulness of this approach, we performed a small preliminary user study. Three retrieval tasks (against the INEX collection, [17]) were given in natural language. Five users performed the tasks with the graphical interface described here, two of them also used a command-line tool to directly enter XIRQL queries. The results indicate that even people with no knowledge of XIRQL are enabled to pose queries using this interface. For more complex queries, the interface might speed up users who know XIRQL. The layout-oriented variant of Query by Example was popular with all users, the DTD-based method was rarely used.

4.2 Result Presentation

The objective of traditional document retrieval systems is to select documents from a collection. For XML documents, the obvious extension is to select parts (elements) of documents, too. Whereas traditional result presentation uses a linear list in order to illustrate the ranking (or the corresponding retrieval status values) this approach does not cover the following two new aspects of XML retrieval:

- Structural relationships: Since different result elements may contain each other, the containment relationship must be visualized.
- Size of answer elements: In classical IR applications, it is assumed that all documents are approximately equal in size. Since XML elements may

contain anything from a few words to a complete article, result element size should be illustrated.

TextBars
treemap
partial treemap

As a typical result set contains elements from several documents, we need a compact representation so that more than one document can be displayed at the same time.

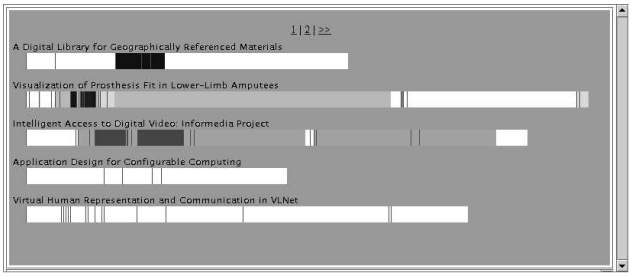

Fig. 3. Result presentation with TextBars.

As a first step in this direction, we developed TextBars, which are based on *TileBars* as described by Hearst [20, 21]. Here, a document is presented as one long bar (where length indicates text size), which is segmented according to the XML structure. The start of an element is indicated with a red line. With this method, only elements present in the result set are shown. Since a retrieval result is always a weighted set, this visualization needs to be extended to deal with the weights, too. For the weights, a visual variable is needed that can be used together with TextBars. Since the weights impose a linear order on the results, the visual variable should be selective (allowing distinction between objects with and without a certain property) as well as ordered (allowing a less-than comparison between values). We choose brightness as the visual variable to use. This is implemented via shades of gray, where white means zero (the object is not relevant at all) and black means one (the object is highly relevant). An example of this visualization method is shown in figure 3.

Whereas TextBars are (in principle) a one-dimensional representation of a document, Treemaps (see Johnson and Shneiderman [22]) use two dimensions in order to illustrate the structure of an XML document (see figure 4 for an example). Here a document is represented as a rectangular area, which is split horizontally for the first level nodes, vertically for the next level again horizontally for the third level, and so on.

However, for XML documents with a rich structure, this representation is too cluttered. Therefore, we augment the concept and introduce *Partial Treemaps*, where we omit nodes in case they are not a retrieved item or an ancestor of a retrieved item (see Kriewel [23]).

Tool-tips provide additional information about each retrieved item. In addition to a list of Partial Treemaps, the document itself is shown (processed

by an XSL style-sheet) together with a 'table of contents' view. The table of contents is a tree view of the document, but certain 'unimportant' XML elements are left out to constrict the size of the tree. The elements to retain are those that contribute to the overall logical structure of the document. (Typically, the `section` element would be included, but the `bold` element would not be included.) As with TextBars, retrieval weights are illustrated as shades of gray. The resulting interface is shown in figure 5.

We performed a small user study to test the effect of the visualization on the time the users needed, and on the quality of the relevance judgments. Five users were given nine queries each, together with a visualization of the query results. Each user chose three queries (query results) for judgment with a textual result representation, and three queries for each of our visualizations. The results indicated that TextBars outperform the textual representation in terms of precision, and partial treemaps improve precison even further. On the other hand, the time used for the judgements were about the same for all three methods; participants reported that they had a closer look at the retrieval results and their relationships when using the graphical methods. Thus, it seems that the added information provided by the graphical method improved the quality of the judgments.

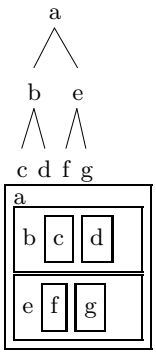

Fig. 4. A simple tree and its treemap.

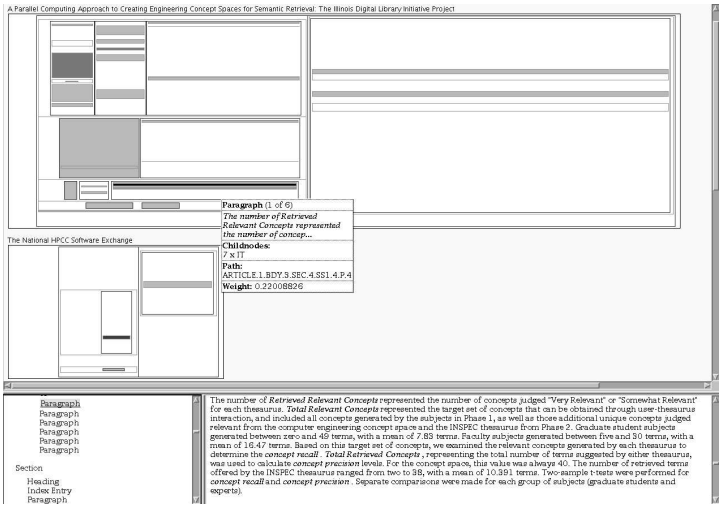

Fig. 5. Result presentation with Partial Treemaps. Each element in the treemap has a tool-tip with a summary about that element. In the bottom left, we show a 'table of contents' (tree showing certain elements) and in the bottom right, we show the document itself, at the spot corresponding to the element in the treemap that the user has clicked on.

5 Related Work

W3C XML Query Working Group
XQuery
XPath
XIRQL
XPath
XIRQL
XPath
structured document retrieval
structured document retrieval!structural approach

Our work presented here is based on the document-centric view of XML, which requires a query language that mainly supports selection based on conditions with respect to both structure and content, taking into account the intrinsic uncertainty and vagueness of content-based retrieval. In contrast, the W3C working group on XML query languages has focused on the data-centric view. Following earlier proposals for XML query languages like XML-QL (Deutsch et al. [11]) or Quilt (Chamberlin et al. [6]), the proposed query language XQuery (Boag et al. [4]) draws heavily on concepts from query languages for object-oriented databases (e.g., OQL) or semistructured data (e.g., Lorel [1]). Due to this origin, XQuery has a much higher expressiveness than XPath and XIRQL. The latter two offer only selection operators, thus results are always complete elements of the original documents. In contrast, XQuery also provides operators for restructuring results as well as for computing aggregations (count, sum, avg, max, min).

A typical XQuery expression has the following structure:

```
FOR PathExpression
WHERE AdditionalSelectionCriteria
RETURN ResultConstruction
```

Here, PathExpression may contain one or more path expressions following the XPath standard, where each expression is bound to a variable. Thus, the FOR clause returns ordered lists of tuples of bound variables. The WHERE clause prunes these lists of tuples by testing additional criteria. Finally, the RETURN clause allows for the construction of arbitrary XML documents by combining constant text with the content of the variables.

Since XIRQL is based on XPath, it can be seen as an extension of a subset of XQuery (i.e., only a `FOR` clause, with a single `PathExpression`) in order to support IR.

The current version of XQuery supports querying for single words in texts only. Recently, discussions about text retrieval extensions for XQuery have started, which aim at providing restricted forms of weighting and ranking (see the requirements [5] and the use cases [2]). However, most of the use cases presented there do not take weighting into account and operate on the syntactical level. (For example, proximity search is required to handle phrases, rather than allowing for linguistic predicates.) Furthermore, ranking is only applicable to full-text search predicates whereas we consider weighting and ranking to be an important feature for other data types, as well, including numbers.

In information retrieval, previous work on structured documents has focused on two major issues:

- The *structural* approach enriches text search by conditions relating to the document structure, e.g., that words should occur in certain parts of a

structured document retrieval!content-based approach
passage retrieval

document, or that a condition should be fulfilled in a document part preceding the part satisfying another condition. Navarro and Baeza-Yates [26] give a good survey on work in this direction. However, all these approaches are restricted to Boolean retrieval, so neither weighting of index terms nor ranking are considered.

- *Content-based* approaches aim at the retrieval of the most relevant part of a document with respect to a given query. In the absence of explicit structural information, passage retrieval has been investigated by several researches (Hearst and Plaunt [19]). Here the system determines a sequence of sentences from the original document that fit the query best.

  Only a few researchers have dealt with the combination of explicit structural information and content-based retrieval. Myaeng et al. [25] use belief networks for determining the most relevant part of structural documents, but allows only for plain text queries, without structural conditions. The FERMI multimedia model (Chiaramella et al. [7]) mentioned before is a general framework for relevance-based retrieval of documents. Lalmas [24] and Fuhr et al. [14] describe refinements of this approach based on different logical models.

Comparing the different approaches described above, it turns out that they address different facets of the XML retrieval problem, but there is no approach that solves all the important issues: The data-centric view as well as the structural approach in IR only deal with the structural aspects, but do not support any kind of weighting or ranking. On the other hand, the content-based IR approaches address the weighting issue, but do not allow for structural conditions.

Only a few researchers have tried to combine structural conditions with weighting. Theobald and Weikum [31] extend XML-QL by weighted document indexing; however, this approach is not based on a consistent probabilistic model. As another approach based on XML-QL, Chinenyanga and Kushmerik [8] introduce an operator for text similarity search on XML documents; so this extension supports only a very specific type of queries. A nice theoretical concept for vagueness with respect to both value conditions and structural conditions is proposed by Schlieder and Meuss [29]; however, the underlying query language is rather restricted.

Recently, several approaches have been suggested for integrating structural and content-oriented conditions in a single query, such as the approach by Piwowarski et al. [28] (based on Bayesian Networks) and Ogilvie and Callan [27] (based on Language Models). However, there is no support for different data types and corresponding search predicates. Grabs and Schek [18] present an approach based on the vector space model which allows binary retrieval on non-text conditions and ranked retrieval on text conditions. This model takes into account contextual term weights. This way, terms that are common in the text of an article, say, but rare in the title, get proper treatment for queries regarding both types of context.

The path algebra approach for processing XIRQL is similar to the proximal nodes model described by Navarro and Baeza-Yates [26]. (The close relationship between XQL and proximal nodes is discussed by Baeza-Yates and Navarro [3].) However, we give a more formal specification of the semantics of the different operators and we also consider hyperlinks. Furthermore, we extend this model by dealing with data types and weighting.

path algebra
XIRQL
vague predicate
XIRQL

6 Conclusions and Outlook

In this paper, we have described a query language for information retrieval in XML documents. Current proposals for XML query languages lack most IR-related features, which are weighting and ranking, relevance-oriented search, data types with vague predicates, and structural relativism. We have presented the new query language XIRQL which integrates all these features, and we have described the concepts that are necessary in order to arrive at a consistent model for XML retrieval.

In order to ease query formulation, we have developed a user interface supporting formulation of syntactically and semantically correct queries. For result presentation of XML retrieval, we have described a solution which visualizes also sizes of result elements and structural relationships between elements.

Based on the concepts described in this paper, we have implemented a retrieval engine named HyREX (*Hy*permedia *R*etrieval *E*ngine for *X*ML). HyREX is designed as an extensible IR architecture. The whole system is open source and can be downloaded from `http://www.is.informatik.uni-duisburg.de/projects/hyrex`. For specific applications, new data types can be added to the system, possibly together with new index structures.

References

[1] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J.L. Wiener. The Lorel query language for semistructured data. *International Journal on Digital Libraries*, 1(1):68–88, May 1997.

[2] S. Amer-Yahia and P. Case. XQuery and XPath full-text use cases. Technical report, World Wide Web Consortium, February 2003. `http://www.w3.org/TR/2003/WD-xmlquery-full-text-use-cases-20030214/`.

[3] R. Baeza-Yates and G. Navarro. XQL and proximal nodes. *Journal of the American Society for Information Science and Technology*, 53(6): 504–514, 2002.

[4] S. Boag, D. Chamberlin, M. F. Fernandez, D. Florescu, J. Robie, and J. Siméon. XQuery 1.0: An XML query language. Technical report, World Wide Web Consortium, 2002. `http://www.w3.org/TR/xquery/`.

[5] Stephen Buxton and Michael Rys. XQuery and XPath full-text requirements. Technical report, World Wide Web Consortium, February 2003. http://www.w3.org/TR/xmlquery-full-text-requirements/.

[6] D. Chamberlin, J. Robie, and D. Florescu. Quilt: An XML query language for heterogeneous data sources. In Suciu and Vossen [30], pages 53–62. ISBN 3-540-41826-1.

[7] Y. Chiaramella, P. Mulhem, and F. Fourel. A model for multimedia information retrieval. Technical report, FERMI ESPRIT BRA 8134, University of Glasgow, April 1996.

[8] T.T. Chinenyanga and N. Kushmerik. Expressive retrieval from XML documents. In Croft et al. [10], pages 163–171.

[9] W. Bruce Croft, Alistair Moffat, Cornelis J. van Rijsbergen, Ross Wilkinson, and Justin Zobel, editors. *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, 1998. ACM.

[10] W.B. Croft, D. Harper, D.H. Kraft, and J. Zobel, editors. *Proceedings of the 24th Annual International Conference on Research and development in Information Retrieval*, New York, 2001. ACM.

[11] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu. XML-QL: A query language for xml. Technical report, World Wide Web Consortium, 1998. http://www.w3.org/TR/NOTE-xml-ql.

[12] David C. Fallside. XML schema part 0: Primer. W3C recommendation, World Wide Web Consortium, May 2001. http://www.w3.org/TR/xmlschema-0/.

[13] N. Fuhr. Towards data abstraction in networked information retrieval systems. *Information Processing and Management*, 35(2):101–119, 1999.

[14] N. Fuhr, N. Gövert, and Th. Rölleke. DOLORES: A system for logic-based retrieval of multimedia objects. In Croft et al. [9], pages 257–265.

[15] N. Fuhr and K. Großjohann. XIRQL: A query language for information retrieval in XML documents. In Croft et al. [10], pages 172–180.

[16] N. Fuhr and T. Rölleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Transactions on Information Systems*, 14(1):32–66, 1997.

[17] Norbert Fuhr, Norbert Gövert, Gabriella Kazai, and Mounia Lalmas, editors. *INitiative for the Evaluation of XML Retrieval (INEX). Proceedings of the First INEX Workshop. Dagstuhl, Germany, December 8–11, 2002*, ERCIM Workshop Proceedings, Sophia Antipolis, France, March 2003. ERCIM. http://www.ercim.org/publication/ws-proceedings/INEX2002.pdf.

[18] T. Grabs and H.-J. Schek. Flexible information retrieval from XML with PowerDB-XML. In Fuhr et al. [17], pages 141–148. http://www.ercim.org/publication/ws-proceedings/INEX2002.pdf.

[19] M.A. Hearst and C. Plaunt. Subtopic structuring for full-length document access. In *Proceedings of the Sixteenth Annual International ACM*

SIGIR Conference on Research and Development in Information Retrieval, pages 59–68, New York, 1993. ACM.

[20] Marti A. Hearst. TileBars: Visualization of term distribution information in full text information access. In *Proceedings of the Conference on Human Factors in Computer Systems, CHI'95*, May 1995.

[21] Marti A. Hearst. User interfaces and visualization. In *Modern Information Retrieval*. Addison Wesley, 1999.

[22] Brian Johnson and Ben Shneiderman. Tree-maps: A space filling approach to the visualization of hierarchical information structures. Technical Report CS-TR-2657, University of Maryland, Computer Science Department, April 1991.

[23] Sascha Kriewel. Visualisierung für retrieval von XML-dokumenten. Master's thesis, University of Dortmund, CS Dept., December 2001.

[24] M. Lalmas. Dempster-shafer's theory of evidence applied to structured documents: Modelling uncertainty. In Nicholas J. Belkin, A. Desai Narasimhalu, and Peter Willet, editors, *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 110–118, New York, 1997. ACM.

[25] S.H. Myaeng, D.-H. Jang, M.-S. Kim, and Z.-C. Zhoo. A flexible model for retrieval of SGML documents. In Croft et al. [9], pages 138–145.

[26] G. Navarro and R. Baeza-Yates. Proximal nodes: a model to query document databases by content and structure. *ACM Transactions on Information Systems*, 15(4):400–435, 1997.

[27] P. Ogilvie and J. Callan. Language models and structure document retrieval. In Fuhr et al. [17], pages 33–40. `http://www.ercim.org/publication/ws-proceedings/INEX2002.pdf`.

[28] B. Piwowarski, G.-E. Faure, and P. Gallinari. Bayesian networks and INEX. In Fuhr et al. [17], pages 149–154. `http://www.ercim.org/publication/ws-proceedings/INEX2002.pdf`.

[29] T. Schlieder and M. Meuss. Result ranking for structured queries against XML documents. In *DELOS Workshop: Information Seeking, Searching and Querying in Digital Libraries*, Sophia Antipolis, France, December 2000. ERCIM. `http://www.ercim.org/publication/ws-proceedings/DelNoe01/`.

[30] Dan Suciu and Gottfried Vossen, editors. *The World Wide Web and Databases: Third International Workshop WebDB 2000, Dallas, Texas, USA, May 18-19, 2000*, volume 1997 of *Lecture Notes in Computer Science*, Heidelberg et al., 2001. Springer. ISBN 3-540-41826-1.

[31] A. Theobald and G. Weikum. Adding relevance to XML. In Suciu and Vossen [30], pages 105–124. ISBN 3-540-41826-1.

Index