Modelling vague content and structure querying in XML retrieval with a probabilistic object-relational framework

Mounia Lalmas and Thomas Rölleke

Department of Computer Science, Queen Mary University of London,
London E1 4NS, England, United Kingdom
{mounia,thor}@dcs.qmul.ac.uk
qmir.dcs.qmul.ac.uk/

Abstract. Many XML retrieval applications require relevance-oriented ranking of retrieved elements in order to capture the vagueness inherent to the information retrieval process. This relevance-oriented ranking should not only support vagueness at the content level, but also at the structural level. In this paper, we use a probabilistic object-relational framework to model representation and retrieval strategies that take into account vagueness at both content and structure level. Our approach makes use of established database technology combined with sound probability theory, thus allowing for fast and flexible prototyping of various representation and retrieval strategies.

1 Introduction

With the adoption of the XML mark-up language on the web and in digital libraries, there is the need to exploit the structural nature of XML documents for the purpose of their retrieval (e.g. [4, 8]). Searching XML documents requires query languages that support selection based on conditions with respect to *both* the content and the structure of XML documents. In addition, these query languages must take into account the intrinsic uncertainty associated with the information retrieval process to allow for a *relevance-oriented* access to XML documents.

A number of query languages for XML retrieval have been or are being proposed (e.g. [1, 7, 3, 5]). These query languages, more precisely, their evaluation, to date, provide an access to content based on string matching and regular expressions, which does not suffice for relevance-oriented search on XML documents. Examples of query languages that provide relevance-oriented search of XML documents include e.g. [9, 18]. However, only the uncertainty associated with the content of the XML documents is considered.

It is well known in information retrieval that users of retrieval systems often find it difficult to express their need for information in form of a query because they can be uncertain about the information they want, or they are unfamiliar with the document collection, thus leading to a vague formulation of the information need. For example, in the context of the world wide web, [11] shows that

62% of queries submitted to Excite were insufficient to yield satisfactory results; these queries did not even contain structure conditions! It is therefore not realistic to assume that users can express with exactitude structure conditions when searching collections of XML documents. Even though some users may correctly express structure conditions, they may prefer to retrieve some types of elements. For example, a user may prefer to retrieve the abstracts before the conclusions of articles, and at last the articles themselves. In addition, some users may want to emphasise content condition versus structure condition, or vice versa. Representation and retrieval strategies that allow for the fact that users may not be able to appropriately express structure conditions, have preferences on the types of elements to be retrieved, and want to emphasise condition types, must consider uncertainty at both content and structure level. That is, they must provide for vague content and vague structure querying.

The INitiative for the Evaluation of XML retrieval (INEX) aims at providing an infrastructure to evaluate the effectiveness of relevance-oriented XML retrieval systems. This is done through the creation of a test collection of real world XML documents along with a set of topics and respective relevance assessments. In INEX 2002 [12], two retrieval tasks were performed: content-only queries, which are information retrieval-style user requests that ignore the document structure; and content-and-structure queries, which are requests that contain explicit references to the XML structure, either by restricting the context of interest or the context of certain search concepts. In INEX 2003[1], a third task was added, vague content-and-structure queries, where the structural constraints of a query can be treated as vague conditions. This shows the importance attached to vague structure querying for relevance-oriented XML retrieval.

In this paper, we describe relevance-oriented representation and retrieval strategies of XML documents, which consider both vague content and vague structure conditions. There are two aspects to our work: first, we introduce the concept of Boolean stemming upon which our vague structure condition is based; and second, we develop our approach for vague content and structure querying, using a probabilistic object-relational framework. The latter provides support for representing complex documents (e.g. XML documents) and describing retrieval strategies (e.g. Boolean stemming) based on generic and well-established approaches to data and information management such as relational database model, logic, and object-orientation.

Making use of established database technologies, in combination to sound probability theory for modelling uncertainty, provides the necessary expressiveness and flexibility for capturing both strict and vague content and structure querying (not just based on our concept of Boolean stemming) for accessing information from large XML repositories. In addition, it yields the flexibility, scalability and expressiveness for supporting the evaluation of the growing family and increasing complexity of XML query languages, as well as adapting to user preferences or task requirements (e.g. the level of vagueness allowed in the structure condition). Therefore, the focus of our paper is to show how our probabilistic

[1] See http://www.is.informatik.uni-duisburg.de/projects/inex03/.

object-relational framework is used to model representation and retrieval strategies that consider vague content and structure querying for a relevance-oriented retrieval of XML documents, based on the concept of Boolean stemming.

The structure of the paper is as follows. In Section 2, we provide a working example that will be used throughout the paper. In Section 3, we present the concept of "Boolean stemming", upon which our vague structure condition is based. In Section 4, we describe the probabilistic object-relational representation of XML documents. In Section 5, we describe the probabilistic object-relational evaluation of XML queries. We conclude and discuss future work in Section 6.

2 Working example

We will use the (very) small XML document and the (fictitious) XPath-based query expression shown below as our working example throughout the paper. Our example document has one title, a section with a title and three paragraphs. The document has a version attribute, with value 1.0. Our query example searches for all paragraph elements in the root document element, where paragraph elements are about "sailing boats".

```
<document version="1.0">
  <title>Hello Sailing World</title>
  <section>
    <title>Sailing on the East Coast Rivers</title>
    <paragraph>Many sailing boats sail at the coast.</paragraph>
    <paragraph>Many sunny sailing days at the coast.</paragraph>
    <paragraph>Fog and drizzle are part of the game.</paragraph>
  </section>
</document>

/document//paragraph[about(., 'sailing boats')]
```

The path component "/document" expresses that the document element should be a child of the root element, and the path component "//paragraph" expresses that the paragraph element should be a descendant of the document element. If the structure condition is treated as strict (i.e. not vague), the above query would be expected to return the first paragraph before the second paragraph (since both, "sailing" and "boats" occur in the first paragraph, whereas only "sailing" occurs in the second paragraph, assuming that stemming has been performed) and no other elements. However, information that is of relevance to the user may also be contained in other elements, although the request was for paragraphs. For example, returning the section may be considered a viable result, since the section is indeed about "sailing boats".

3 "Boolean stemming"

An XML query expression is formed by the combination of structure and content conditions. In our query example, "/document//paragraph" corresponds to the

structure condition, whereas "about(.,'sailing boats')" corresponds to the content condition. The common approach is to interpret the query as a *conjunction* of the structure condition and the content condition. However, from information retrieval, we know that a conjunction of conditions is precision- rather than recall-oriented, i. e. we increase the likelihood of retrieving a higher portion of relevant documents (elements), but we increase the likelihood of missing relevant documents (elements). We also know that, from a ranking point of view, a document (an element) that fulfils many conditions will be ranked higher than a document (an element) that fulfils less conditions. It makes therefore sense to view the combination of structure and content conditions as a *probabilistic disjunction* of conditions. Rather than using the above *conjunctive* query, we formulate the following *disjunctive* query:

```
/document//paragraph $or$ //*[ about(., 'sailing boats')]
```

The same transformation can be applied to the structure condition. The path expression "/document//paragraph" is usually interpreted as a conjunction of path sub-expressions: *The element to be retrieved shall be a paragraph AND its ancestor element shall be a document AND the ancestor element shall be a child of the root element'*. By replacing the conjunction of path sub-expressions by a disjunction of path sub-expressions, we obtain the following interpretation: *The element to be retrieved shall be a paragraph OR its ancestor element shall be a document OR the ancestor element shall be a child of the root element*.

We refer to this mapping of a conjunctive expression to a disjunctive expression as *Boolean stemming*. We use the term "stemming" because we apply rules for replacing AND by OR. In our example, we used the most general rule, namely, "replace each occurrence of AND by OR". More specific rules can also be applied, for example, "replace the last occurrence of AND by OR", thus allowing for specifying various levels of vagueness as introduced later in this paper.

The replacement of AND by OR increases recall (i.e. the number of retrieved and relevant elements) at the costs of retrieving more non-relevant elements, which can have an impact on effectiveness and efficiency. With respect to effectiveness, a ranking function would ensure that the best elements are retrieved first, while at the same time allowing for vague characterisation of the structure condition. The fact that we retrieve more elements will decrease efficiency. However, we avoid join operations and complex regular expression matches that would be required for the evaluation of conjunctions.

4 Probabilistic object-relational representation of XML documents

In this section, we describe how XML documents are represented within our probabilistic object-relational framework. The representation involves two steps. XML documents are first translated into POOL programs (Section 4.1), which are then translated into PRA expressions (Section 4.2) following the so-called object-relational approach.

4.1 POOL representation of XML documents

POOL (which stands for probabilistic object-oriented logic) is a probabilistic representation and retrieval framework designed for representing and retrieving complex objects (see [15]). It can be used for describing the retrieval of any type of structured document standard (e.g. HTML, XML, MPEG-7). Particular to POOL is the integration of probability theory and the object-oriented paradigm. Next is an abbreviated syntax of POOL (more information can be found in [13]).

program	::= clause program
clause	::= fact | context-clause | query | rule
fact	::= proposition | probability proposition
proposition	::= term | classification | relationship
term	::= NAME
classification	::= NAME '(' NAME ')'
relationship	::= NAME '.' NAME '(' NAME ')'
context-clause	::= context | probability context
context	::= NAME '[' program ']'
query	::= '?-' subgoal-list
subgoal-list	::= subgoal | subgoal '&' subgoal-list
subgoal	::= fact-subgoal | context-subgoal
fact-subgoal	::= atom | NOT atom
atom	::= term-atom | classification-atom | relationship-atom
context-subgoal	::= context-name '[' subgoal-list ']'
rule	::= goal ':-' subgoal-list
goal	::= atom

Below we show an excerpt of the POOL program corresponding to our example document.

```
document(document1)
version.document1(1.0)
document1 [
  title(title1)
  section(section1)
  title1 [ 0.8 hello sail 0.3 world ]
  section1 [
     title1 [ sail east coast river ]
     0.2 paragraph1 [ 0.7 sail  0.4 coast  0.8 boat ]
     0.5 paragraph2 [ 0.2 sunny 0.8 sail 0.2 day 0.5 coast ]
     0.5 paragraph3 [ 0.5 fog 0.2 drizzle 0.5 part 0.3 game ] ] ]
```

"document1", "title1", etc. correspond to contexts in POOL, and model the elements "document[1]", "title[1]", etc., respectively. To express the parent-child structure between elements, POOL uses square brackets to delimit the contexts modelling the elements. For example, "section1" is a sub-context of "document1" ("section[1]" is a child element of "document[1]"), and hence, appears within

"document1" context. Classifications provide the class (element) type of each element, e.g. "document", "title", "section", etc. Relationships provide attributes and their value for the elements (contexts).

Particular to POOL is the consideration of two types of uncertainty. Probabilities can be assigned at the content level (in front of a proposition) to represent how good a term[2] is at describing the content of an element; and at the structure level (in front of a context) to represent the impact an element has towards describing the content of its parent element. The probabilities assigned to "sail", "coast", "boat", etc. in a context describe how well these terms describe the content of the element modelled by that context. For example, the term "sail" is a better representation of "document[1]/section[1]/paragraph[1]" than the term "coast". These probabilities can be derived using standard term frequency (tf) information. In our example, the probabilities are fictitious (our XML document is too small to produce meaningful probability values). A probability value of 1.0 is assumed when no probability value is explicitly given.

Of particular interest to XML retrieval are the structure probabilities. In XML retrieval, the content of a parent element is often characterised as the aggregation of its content and the content of its child elements (see [8] for some examples). However, not all child elements will have the same impact in contributing to the content of the parent element. For example, an abstract may be a better reflection of what an article is about than a conclusion. This impact is modelled by assigning probabilities at the structure level. For example, in our example, "paragraph2" has a higher impact than "paragraph1" towards the content of "section1". There are no standard mechanisms to compute the structure probabilities yet: user studies could be performed to elicitate these values, extensive retrieval experiments can be carried out to empirically determine optimal values, or machine learning methods could be used. This is currently a research issue.

4.2 PRA representation of XML documents

For evaluating POOL programs, we translate them into PRA (which stands for probabilistic relational algebra) expressions. PRA expressions representing XML documents consist of probabilistic relations, i.e., relations to which probability values are attached. In our framework, the representation of XML documents for the purpose of relevance-oriented retrieval makes use of the following four relations:

1. *instance_of(element-id, class)*: representing the type (class) of an XML element.
2. *part_of(child-id, parent-id)*: representing the child-parent structure between two XML elements.
3. *attribute(element-id, attribute-name, attribute-value)*: representing an attribute together with its value for an XML element.

[2] Probabilities can also be assigned to classifications and relationships. For simplicity, these cases are not considered in this paper.

4. *term(term, element-id)*: representing the content of (the terms appearing in) an XML element.

The first three relations are motivated by three of the basic pillars of object-oriented modelling, namely classification, aggregation (which we use to model the structure of XML documents) and relationships (which we use to model the attributes of XML documents). The fourth relation reflects the content dimension requires to perform relevance-oriented retrieval. The translation of our POOL program example into the above four relations is shown below:

```
instance_of(element-id, class):
(/document1, document)
(/document1/title1, title)
(/document1/section1, section)
(/document1/section1/title1, title)
(/document1/section1/paragraph1, paragraph)
(/document1/section1/paragraph1, paragraph)
(/document1/section1/paragraph1, paragraph)

part_of(child-id, parent-id):
(/document1/title1, document1)
(/document1/section1, document1)
(/document1/section1/title1, /document1/section1)
(/document1/section1/paragraph1, /document1/section1)
(/document1/section1/paragraph1, /document1/section1)
(/document1/section1/paragraph1, /document1/section1)

attribute(element-id, attribute-name, attribute-value):
(/document1, version, 1.0)

term(term, element-id):
(east, /document1/section1/title1)
(coast, /document1/section1/title1)
(sailing, /document1/section1/paragraph1)
(boats, /document1/section1/paragraph1)
...
```

The content and structure probabilities in our POOL program are associated to the "term" and the "part_of" relations, as shown below. A probability value of 1.0 is assigned to all tuples of the "instance_of" and "attribute" relations, so these relations are not shown.

term	
Prob	Tuple
1.0	(sail, /document1/title1)
0.7	(sail, /document1/section1/paragraph1)
0.4	(coast, /document1/section1/paragraph1)
0.8	(boat, /document1/section1/paragraph1)
0.8	(sail, /document1/section1/paragraph2)
...	...

termspace	
Prob	Tuple
0.562	(boat)
0.492	(east)
0.289	(coast)
0.097	(sail)
...	...

part_of	
Prob	Tuple
1.0	(/document1/title1, /document1)
1.0	(/document1/section1, /document)
0.2	(/document1/section1/paragraph1, /document1/section1)
0.5	(/document1/section1/paragraph2, /document1/section1)
...	...

classpace	
Prob	Tuple
1.0	(section)
1.0	(document)
0.64	(title)
0.43	(paragraph)

Other relations are used. For example, considering inverse document frequency (idf) together with term frequency is known to increase retrieval effectiveness. In the context of XML retrieval, considering the inverse element frequency of a term allows for discriminating between relevant and non-relevant elements. Indeed, a term that occurs in all elements, i.e has a low inverse element frequency, is not useful for describing the content of a particular element since it cannot discriminate that element from others. The inverse element frequency values in a collection are represented by a probabilistic relation. The above figure shows a relation called "termspace", which displays (the probabilistic interpretation of) the inverse element frequency values for our example program; the highest the probability, the better the term is at discriminating between relevant and non-relevant elements.

Another relation is defined to represent the importance associated to element types. The relation "classpace" above shows an example of such a relation. As for the structure probabilities, there is no standard way to obtain the type probabilities, apart from either eliciting them from users and user studies, or through extensive experimentation. In our work, and our example, we adopt the following strategy: the higher the occurrence frequency of an element type, the lowest its probability.

To recap, XML documents are transformed into POOL programs, where content, structure, element type and classification TOGETHER with their associated uncertainty are represented. POOL programs are then translated into probabilistic relations, which capture any aspect deemed necessary for the vague content and structure querying of XML documents. In the next section, we describe how the same framework is used to evaluate XML queries using these representations.

5 Probabilistic object-relational evaluation of XML queries

As for the representation of XML documents, the evaluation of XML queries is also performed in two steps. First, XML queries are translated into POOL

programs (Section 5.1). It is during this translation that Boolean stemming is applied. Then POOL programs are translated into PRA expressions (Section 5.2), which access the probabilistic relations corresponding to the PRA representations of XML documents to perform retrieval.

5.1 POOL evaluation of XML queries

XML queries are translated into POOL queries, taking into account a vagueness level, which can be specified by a user, or set by the system for a particular application. Our current implementation allows three levels of vagueness.

vagueness level	explanation
0	no Boolean stemming
1	use a disjunction for combining structure and content criteria
2	use a disjunction for combining the path sub-expressions

Our example query is translated, using vagueness level 0, to the following POOL query:

```
retrieve(X) :- document(X) & /X[paragraph(Y) & //Y[sail]]
retrieve(X) :- document(X) & /X[paragraph(Y) & //Y[boat]]
?- retrieve(X)
```

Elements (contexts) to be retrieved are of class "paragraph", are children of the root context of class "document", and contain the terms "sail" or "boat". The first "retrieve" rule corresponds to the "sail" content condition, and the second to the "boat" content condition. Vagueness level 2 is implemented by dropping the nesting of elements and by ignoring "/" and "//". Our POOL query, using for vagueness level 2 is:

```
retrieve(X) :- document(X)
retrieve(X) :- paragraph(X)
retrieve(X) :- X[ sail ]
retrieve(X) :- X[ boat ]
?- retrieve(X)
```

Elements (contexts) to be retrieved are of class "document" or "paragraph", or contain the terms "sail" or "boat".

5.2 PRA evaluation of XML queries

PRA evaluation of XML queries consist of sequences of probabilistic-based standard relational algebra operations (Union, Join, Project, Select, Subtract). As described in the previous section, probability values are attached to tuples (e.g. "term" tuples, "part_of" tuples, "termspace" tuple, etc.). PRA expressions access these tuples, and their probabilities are combined to produce other probabilistic relations, until returning the retrieval results themselves, also probabilistic relations. For example, in the Project operation, when probabilistic independence is assumed among the tuples of a relation, the probabilities of duplicate

tuples are added (see [10] for details). In this section, we illustrate the use of PRA to implement various retrieval strategies of the our POOL query with vagueness level 2. As an introduction, we first start with a simple POOL (content-only) query, which requests for document about "sail".

```
?- X[ sail ]
```

There are two ways to translate the above POOL query into a PRA query, depending on which relations, i.e. "term" and/or "termspace" will be accessed. The first PRA expression below uses the "term" relation only, i.e. considers term frequency information only; the second uses the two relations, i.e. considers term frequency and inverse element frequency information ($i refers to the ith column of the relation).

```
retrieve1 =  Project[$2](Select[$1=sai](term))
retrieve2 =  Project[$3](Join[$1=$1](Select[$1=sail](termspace), term))
```

In "retrieve1", selection is first applied on the "term" relation with respect to the term "sail", then a projection is applied to obtain the relevant elements. In "retrieve2", after a selection with respect to the term "sail", a join operation is performed between the "term" and the "termspace" relations (this is the equivalent to a probabilistic interpretation of the standard tf $\times$ idf in information retrieval), and then the projection is performed to obtain the retrieved elements. The retrieved elements for our XML document are shown below for the above two PRA expressions:

retrieve1 (term)	
Prob	Tuple
1.0	(/document1/title1)
1.0	(/document1/section1/title1)
0.8	(/document1/section1/paragraph2)
0.7	(/document1/section1/paragraph1)

retrieve2 (term and termspace)	
Prob	Tuple
0.097	(/document1/title1)
0.097	(/document1/section1/title1)
0.077	(/document1/section1/paragraph2)
0.068	(/document1/section1/paragraph1)

In "retrieve1", the two titles are retrieved with probability 1.0 since this is the probability assigned to "sail" in the two titles. Accordingly, the paragraphs are retrieved with probabilities 0.8 and 0.7, the probabilities assigned to "sail" in the paragraphs. In "retrieve2", the probabilities are shown for the case that the termspace probability of "sail" (0.097) is multiplied with the probability of "sail" in the elements. Though the ranking remains the same for a query with one term, the ranking might change significantly for a query with several terms. For example, a query about "sailing coast" based on an evaluation strategy without termspace retrieves the titles with 1.0 and "paragraph2" with 0.5. Considering

the termspace probabilities, titles are retrieved with probability 0.097, whereas "paragraph2" is retrieved with $0.5 \cdot 0.289 \approx 0.145$; this results changes the order of retrieved elements.

Our working example POOL query with vagueness level 2 can be translated into several PRA expressions, depending on which relations described in Section 4.2 are used, which itself depends on which retrieval strategy is adopted. A translation using the "term" relation to refer to content, and to "instance_of" to refer to structure yields the following PRA query[3]:

```
retrieve_document = Project[$1](Select[$2='document'](instance_of))
retrieve_paragraph = Project[$1](Select[$2='paragraph'](instance_of))
retrieve_sail = Project[$2](Select[$1='sail'](term))
retrieve_boat = Project[$2](Select[$1='boat'](term))
retrieve = Unite(retrieve_document, retrieve_paragraph,
                 retrieve_sail, retrieve_boat)
```

"retrieve_document" selects all elements of "document" type; "retrieve_paragraph" selects all elements of "paragraph" type; "retrieve_sail" selects all elements containing the term "sail"; and similarly for "retrieve_boat". In all four relations, projection is applied to obtain the elements, which are then all united to provide the final ranking.

To consider inverse element frequency information and class frequency information, join operations are performed between the "term" and the "termspace" relations, and between the "instance_of" and the "classpace" relations, respectively. For example, the PRA equations for "retrieve_document" and "retrieve_sail" for the above PRA query are replaced by the following two PRA equations, respectively:

```
retrieve_document_classpace =  Project[$2](
       Join[$1=$2](Select[$1='document'](classpace), instance_of))

retrieve_sail_termspace = Project[$3](
       Join[$1=$1](Select[$1='sail'](termspace), term))
```

The retrieved elements for our XML document are shown below:

retrieve	
Prob	Tuple
1.000	(/document1)
0.711	(/document1/section1/paragraph1)
0.479	(/document1/section1/paragraph2)
0.435	(/document1/section1/paragraph3)
0.097	(/document1/section1/title1)
0.097	(/document1/title1)

"document1" is retrieved with 1.0 since the document element matches the vague interpretation of the structure condition, and the type "document" is rare

[3] The actual translation leads to a more complex expression, because translation rules are generic, and optimisation is performed. All PRA expressions in this Section are simplified for clarity.

in our small XML example. Element "paragraph1" is ranked higher than "paragraph2" since "paragraph1" contains both "sail" and "boat", whereas "paragraph2" contains only "sail", although the probability of "sail" is higher in "paragraph2" than in "paragraph1". "paragraph3" is retrieved merely because of the structure condition to retrieve paragraphs (probability of "paragraph" in "classpace" is 0.43). The titles are retrieved merely because of the content condition; here, the probability of "sail" in "termspace" leads to the element probability of 0.097, as explained above.

The above ranking illustrates a major problem: "document1" and "paragraph3" are retrieved because of the structure condition, but their retrieval probabilities are comparatively high. This is due to the probabilities in the "classpace" relation. A disjunction which assumes content and structure condition to be independent as done so far, will highly rank an element that matches one condition (structure or content) to a high degree. This leads directly to the requirement that we would like to control the degree to which the match of structure or content condition contributes to the final retrieval probabilities.

Therefore, we consider now preferences on the structure and content condition in the so-called "conditionspace" relation. Whereas we showed clear criteria for estimating the probabilities of the "termspace" and "classpace" relations, it is currently still an open question of how to estimate the probabilities of the "conditionspace" relation. In the scope of this paper, we do not discuss further the estimation, and we assume that probabilities regarding which types of conditions is important to a particular group of users is that given below. The corresponding ranking is also shown below. In our case, our group of users puts more emphasise to the content condition than the structure condition.

conditionspace	
Prob	Tuple
0.6	content
0.4	structure

retrieve	
Prob	Tuple
0.400	(/document1)
0.416	(/document1/section1/paragraph1)
0.212	(/document1/section1/paragraph2)
0.174	(/document1/section1/paragraph3)
0.058	(/document1/section1/title1)
0.058	(/document1/title1)

This result and the overall strategy with "termspace", "classpace" and "conditionspace" has now the desired properties and flexibility for modelling vague structure and content retrieval, although in this paper, for simplicity, we did not take into account the POOL probabilities that an element contributes to the content of its parent element (see [16] for this). For the result in this paper, we observe that "document1" is retrieved now with a smaller probability than "paragraph1". Although the class "document" has a maximal probability in our example, "paragraph1" is ranked higher since "paragraph1" fulfils the structure and the content conditions.

The PRA expression for joining the "conditionspace" relation with the retrieval strategy based on the "classpace" and "termspace" relations is rather complex. The overall idea is that a condition attribute column is added to the relation "retrieve" and that the following join is performed:

```
retrieve = Project[$3, 'structure'](Join[$1=$2]
            (Select[$1='document'](classpace), instance_of))
cond_retrieve = Project[$2](Join[$1=$2](conditionspace, retrieve))
```

In this paper, we do not extend further on the probabilistic assumption underlying the PRA operations. Here, we assumed independence throughout, where for the "conditionspace" relation it seems appropriate to argue a disjointness (weighted sum) assumption rather than the independence assumption applied here . This is also reflected by the choice of probabilities in the "conditionspace" relation, i.e. they sum up to 1.0. Where independence and where disjointness assumption is the better choice will be investigated experimentally (but see [14] for some initial theoretical investigation on this).

6 Summary and Outlook

A relevance-oriented ranking of XML documents must acknowledge the fact that not all users express (or retrieval tasks can have) clear structure conditions, and that users may have preferences, e.g. the types of elements to retrieve. Therefore, uncertainty must be taken into account, not only at the content level, but also at the structure level in relevance-oriented XML retrieval applications.

In this paper, we described representation and retrieval strategies that model uncertainty at both content and structure level. We used a probabilistic object-relational framework for this purpose, thus allowing fast and flexible prototyping of various representation and retrieval strategies in a uniform, comparable and elegant manner. Our approach is based on probability theory and object-relational modelling principles where neither the probability estimation, nor the relations are pre-defined. Therefore, the level of Boolean stemming, the approach we use to provide for vague structure condition, and the estimation of probabilistic parameters such as "termspace", "classpace", and "conditionspace" can be adapted according to user preferences or specific task requirements.

Our next step is to evaluate the different representation and retrieval strategies in terms of effectiveness and efficiency. This will done as part of the INEX initiative, which aims at providing an infrastructure to evaluate the effectiveness of relevance-oriented XML retrieval systems, through the creation of a test collection of XML documents [12].

Other work allowing for vague structure querying is that of e.g. [17, 6, 2], which can be briefly summarised as partial (uncertain) matches between the document and the query structure conditions using various simple to more complex strategies. We intend to model them in our object-relational probabilistic framework thus comparing the effectiveness and efficiency of our Boolean stemming to theirs.

References

1. S. Abiteboul, D. Quass, J. McHugh, J. Widom, J. Wiener, and J. Widom. The Lorel Query Language for Semistructured Data. *International Journal on Digital Libraries*, 1(1):68–88, April 1997.

2. S. Amer-Yahia, S. Cho, and D. Srivasta. Tree pattern relaxation. In *Proceedings of International Conference on Extended Database Technology (EDBT)*, 2002.
3. A. Berglund, S. Boag, D. Chamberlin, M.F. Fernandez, M. Kay, J. Robie, and J. Simeon. XML Path Language (XPath) 2.0. W3C Working Draft, November 2002. http://www.w3.org/TR/xpath20.
4. H. Blanken, R. Schenkel T. Grabs, and G. Weikum, editors. *Intelligent Search on XML*. Springer-Verlag, 2003.
5. S. Boag, D. Chamberlin, M.F. Fernandez, D. Florescu, J. Robie, and J. Simeon. XQuery: An XML query language. W3C Working Draft, 2002. http://www.w3.org/TR/XQuery.
6. D. Carmel, Y.S. Maarek, M. Mandelbrod, Y. Mass, and A. Soffer. Searching XML documents via XML fragments. In *Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 151–158, Toronto, Canada, July 2003.
7. D. Chamberlin, J. Robie, and D. Florescu. Quilt: An XML query language for heterogeneous data sources. In *International Workshop on the Web and Databases (WebDB)*, pages 53–62, Texas, USA, May 2000.
8. N. Fuhr, N. Goevert, G. Kazai, and M. Lalmas, editors. *INEX: Evaluation Initiative for XML retrieval - INEX 2002 Workshop Proceedings*, DELOS Workshop, 2003.
9. N. Fuhr and K. Grossjohann. XIRQL: A query language for information retrieval in XML documents. In *Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval*, New Orleans, USA, August 2001.
10. 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.
11. B.J. Jansen, A. Spink, and T. Saracevic. Real life, real users and real needs: A study and analysis of user queries on the web. *Information Processing & Management*, 36(2):207–227, 2000.
12. G. Kazai, M. Lalmas, N. Fuhr, and N. Goevert. A report on the first year of the INitiative for the Evaluation of XML Retrieval (INEX02). *Journal of the American Society for Information Science and Technology*, 2004. In press.
13. M. Lalmas, T. Rölleke, and N. Fuhr. Intelligent hypermedia retrieval. In P. S. Szczepaniak, F. Segovia, and L. A. Zadeh, editors, *Intelligent Exploration of the Web*. Springer-Verlag Group (Physica-Verlag), 2002.
14. T. Roelleke. A frequency-based and a poisson-based probability of being informative. In *Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 227–234, Toronto, Canada, July 2003.
15. T. Rölleke. *POOL: Probabilistic Object-Oriented Logical Representation and Retrieval of Complex Objects*. Shaker Verlag, Aachen, 1999. Dissertation.
16. T. Rölleke, M. Lalmas, G. Kazai, I. Ruthven, and S. Quicker. The accessibility dimension for structured document retrieval. In *Proceedings of the BCS-IRSG European Conference on Information Retrieval (ECIR), Glasgow*, March 2002.
17. T. Schlieder and M. Meuss. Result ranking for structured queries against xml documents. In *DELOS Workshop: Information Seeking, Searching and Querying in Digital Libraries*, Zurich, Switzerland, 2000.
18. A. Theobald and G. Weikum. The index-based XXL search engine for querying XML data with relevance ranking. In *Advances in Database Thechnology - EDBT 2002, 8th International on Extending Database Technology*, pages 477–495, Prague, Czech Republic, March 2002.