XML Information Retrieval

Mounia Lalmas
University of Glasgow
mounia@acm.org

Abstract

Nowadays, increasingly, documents are marked-up using XML, the format standard for structured documents. In contrast to HTML, which is mainly layout-oriented, XML follows the fundamental concept of separating the logical structure of a document from its layout. This document logical structure can be exploited to allow a focused access to documents, where the aim is to return the most relevant fragments within documents as answers to queries, instead of whole documents. This article describes approaches developed to query, represent, and rank XML fragments.

Keywords: XML, structured document, logical structure, element, focused retrieval, query language, representation strategy, ranking strategy, information retrieval

1 Introduction

Documents can be structured or unstructured. Unstructured documents have no (or very little) fixed pre-defined format, whereas structured documents are usually organized according to a fixed pre-defined structure. An example of a structured document is a book organized into chapters, each with sections made of paragraphs and so on. Nowadays, the most common way to format structured content is with the W3C standard for information repositories and exchanges, the eXtensible Mark-up Language (XML)[1].

Much of the content available on the web is formatted in HTML. Although HTML imposes some structure on a web content, this structure is mainly for presentation purposes and carries little meaning. In contrast, XML is used to provide meaning about the stored content. More precisely, in the context of text documents, with which this article is concerned, XML is used to specify the logical, or tree, structure of documents, in which separate document parts (e.g. chapter, section, abstract) and their logical structure (e.g. a chapter made of sections, a section and its title, an article and its abstract) are explicitly marked-up. As an increasing number of documents are being made available in XML format, effective means to access them are needed. As for standard (unstructured) documents, this requires appropriate query languages, representation methods, and ranking algorithms.

[1] www.w3.org/XML/

Approaches for accessing logically structured documents were first proposed in the 1990s (e.g. [51, 12, 19, 72]). In the late 1990s, as XML was adopted as the standard document format, approaches for what became known as XML information retrieval were being developed (e.g. [15, 6, 10]). Research in XML information retrieval was then further boosted with the set-up in 2002 of the Initiative for the Evaluation of XML Retrieval (INEX) [32], a yearly evaluation campaign that provides a forum for the evaluation of approaches specifically developed for XML information retrieval. INEX provides test collections and evaluation measures, which make it possible for organizations worldwide to evaluate and compare their XML information retrieval approaches.

By exploiting the logical structure of XML documents, the goal of an XML information retrieval system is to implement so-called focused retrieval strategies, which aim at returning document components, i.e. XML elements, instead of whole documents in response to a user query. These focused retrieval strategies aim to break away from the traditional retrieval unit of a document as a single large (text) block. This is believed to be of particular benefit for information repositories containing long documents, or documents covering a wide variety of topics (e.g. books, user manuals, legal documents), where the users effort to locate relevant content within a document can be reduced by directing them to the most useful parts, i.e. the most useful XML elements, in the document.

To identify the most useful XML elements to return as answers to given queries, XML information retrieval systems require:

- Query languages that allow users to specify the nature of relevant components, in particular with respect to their structure.
- Representaion strategies providing a description not only of the content of XML documents, but also their structure.
- Ranking strategies that determine the most relevant elements and rank these appropriately for a given query.

In this article, we provide an overview of query languages (Section 2), representation strategies (Section 3), and ranking strategies (Section 4) developed for XML information retrieval. The representation and ranking strategies presented in this article were evaluated within the INEX evaluation campaigns [22, 25, 27, 26, 28, 24]. The article finishes with some conclusions on XML information retrieval research, and some references to early work related to XML information retrieval (Section 5).

2 Query languages

XML documents are organised into a logical structure, as provided by the XML markup. For example, a scientific article, such as those forming the IEEE test collection used in INEX (see Figure 1), consists of a front matter (<fm>), a body (<bdy>), and a back matter (<bm>). The front matter contains the article's metadata, such as title, author, publication information, and abstract. Following it is the article's body, which contains the actual content of the articles, and is structured into sections (<sec>), subsections (ss1>), and sub-sub-sections (<ss2>). These logical units start with a title,

followed by a number of paragraphs. The back matter contains a bibliography and further information about the article's authors.

Users may want to specify conditions to limit the search to specific XML elements. For example, a user may want sections discussing "XML retrieval evaluation", whereas another user may look for paragraphs about "effectiveness measures" contained in sections about "XML retrieval evaluation". Here we have a combination of content constraints, "XML retrieval evaluation" and "effectiveness measures", typical to information retrieval, and structural constraints, "section", "paragraph" and "paragraph within section". XML query languages have been developed with the aim to express various levels of content and structural constraints. They can be classified as content-only or content-and-structure query languages.

Content-only queries make use of content constraints only, i.e. they are made of words, which is the standard form of input in information retrieval. They are suitable for XML retrieval scenarios where users do not know, or are not concerned, with the document logical structure when expressing their information needs. Although only content conditions are being specified, XML information retrieval systems must still determine what are the best fragments, i.e. the XML elements at the most appropriate level of granularity, to return to satisfy these conditions. For example, the best answer for a query "XML retrieval evaluation" may be a sub-section and not a section, as the section, although relevant, may be less specific to the query than the sub-section. An XML information retrieval system task is to determine this appropriate level of granularity for any given query.

Content-and-structure query languages provide a means for users to specify content and structural information needs. It is towards the development of this type of queries that most research on XML query languages lies. We can distinguish between three main categories of content-and-structure XML query languages, namely tag-based languages (Section 2.1), path-based languages (Section 2.2), and clause-based languages (Section 2.3). For the latter two types, we provide a brief description, mainly through examples, of current languages, namely XPath and NEXI, and XQuery and XQuery Full-Text, respectively.

2.1 Tag-based queries

With tag-based queries, words in the query are annotated with a single tag name, which specifies the type of desired result elements, e.g. a section, an abstract. For example, the information need "retrieve sections about XML retrieval evaluation" would be expressed as `section:XML retrieval evaluation`. An example of a tag-based query language is XSEarch [20].

Tag-based queries are intuitive, and have been used in domains outside XML information retrieval (e.g. faceted search, web search). However they only express simple, although important and likely common, structural constraints. They cannot express, for instance, relationship (structural) constraints, e.g. "a paragraph contained in a section", which may be needed for complex retrieval scenarios.

```
<article>
  <fm>
  ...
    <ti>IEEE Transactions on ...</ti>
     <atl>Construction of ...</atl>
     <au>
       <fnm>John</fnm>
       <snm>Smith</snm>
       <aff>University of ...</aff>
     </au>
     <au>...</au>
  ...
   </fm>
   <bdy>
     <sec>
       <st>Introduction</st>
         <p>...</p>
        ...
      </sec>
      <sec>
       <st>...</st>
       ...
       <ss1>...</ss1>
       <ss1>...</ss1>
      ...
    </sec>
    ...
  </bdy>
  <bm>
    <bib>
     ...
    </bib>
</bm>
</article>
```

Figure 1: Sketch of the structure of a typical article in the INEX test collection

2.2 Path-based queries

Path-based queries are based upon the syntax of XPath (XML Path language[2]), which has been defined by the W3C to navigate to components of an XML document. The most important concept in XPath is the location path, which consists of a series of navigation steps characterising movements within an XML document.

For example, `chapter/section` is a location path, where `chapter` and `section` are steps that navigate to elements of types "chapter" and "section", respectively. The fact that the steps are separated by "/" means that the location path selects `section` elements directly below `chapter` elements. Section elements are referred to as children of chapter elements. The navigation steps can be separated by "//". For example, `chapter//section` navigates to all section elements that are directly or indirectly below a chapter element. Section elements are referred to as descendants of chapter elements. Special steps include the self step denoted "." and parent step "..". For example, `.//section` returns all section elements contained directly or indirectly in the currently navigated element.

At each step, predicates can be specified between "[" and "]", which must be satisfied for elements to be navigated into. For example, the following XPath query `//article[@year=2002]/title` selects the "titles" of "articles" published in 2002, and only those.

An important function in XPath for the purpose of XML information retrieval is the function `contains()`. For example, the query `//section[fn:contains(./title, "XML retrieval")]` will return all section elements with a title containing the string "XML retrieval". The result of this XPath query is a set of section elements, and not a ranked list of section elements. Thus XPath is not an XML query language that can be directly used in XML information retrieval. Nonetheless, it is used by, or has inspired, other path-based query languages, some of which allowing the ranking of results, e.g. XXL [63], XIRQL [23], and NEXI [69]. We discuss the last one, NEXI.

The Narrowed Extended XPath I (NEXI) query language was developed by INEX, as a simple query language for XML information retrieval evaluation. NEXI consists of a small but enhanced subset of XPath. The enhancement comes from the introduction of a new function, named `about()`, which requires an element to be about some specified content criteria. It replaces the XPath `contains()` function, to reflect that an element can be relevant to a given query without actually containing any of the words used in the query.

A small subset of XPath was chosen because NEXI was not developed to test the expressiveness of a query language for XML information retrieval, but to evaluate XML information retrieval effectiveness. For instance, the parent/child navigation step "/" was considered particularly problematic as it was open to misinterpretation by assessors[3], and hence was dropped. Also, all result elements must have at least one `about()` function. This is because for the purpose of evaluating retrieval effectiveness, what matters is that the relevant elements are actually returned. For instance, the following query `//section[about(., XML retrieval)]//title`, which

[2] www.w3.org/TR/xpath

[3] In information retrieval evaluation, assessors are used in the process of building a test collection. Their task is to judge the relevance of the information returned to them as answers to given queries.

requests titles of sections about "XML retrieval evaluation", is not allowed in NEXI; it is indeed a mechanical process to return the title of a section deemed relevant to "XML retrieval".

We finish with an example of a NEXI query:

```
//article[about(.//bdy, XML retrieval)]//
                    section[about(., evaluation)]
```

This query is asking for section elements about "evaluation" contained in articles that have a body that discusses "XML retrieval".

NEXI was developed by INEX for the purpose of evaluating XML information retrieval effectiveness. It remains the task of the XML information retrieval system to interpret a NEXI query, where the interpretation is with respect to the `about()` condition as implemented by the retrieval model, and the structural constraint as implemented by the query processing engine, used by the XML information retrieval system. Sections 4.1 and 4.2 describe approaches used to implement the `about` conditions, whereas Section 4.3 describes approaches used to process structural constraints, for the purpose of ranking XML elements for given queries.

2.3 Clause-based queries

Clause-based queries for XML information retrieval can be compared to SQL, the standard query language for (relational) databases. These queries are made of nested clauses to express information needs. The most prominent clause-based query languages for XML information retrieval are XQuery[4] and XQuery Full-Text[5].

XQuery is an XML query language that includes XPath as a sub-language, but adds the possibility to query multiple documents and combine the results into new XML fragments. The core expressions of XQuery are the FLWOR expressions, which we illustrate with an example. The following query is a FLWOR expression that lists the authors, ordered by their last name, that have written at least 100 articles:

```
for $aut in (doc("aut.xml")//author)
 let $c :=
    count(doc("article.xml")/article[author=$aut])
       where $c>100
       order by $aut/lastname
       return
           <author> { $aut/lastname , $c } </author>
```

The `for` (F in FLWOR) clause binds the variable $aut so that it iterates over the author elements in the document "aut.xml" in the order that they appear. For every such binding, the `let` (L) clause binds the variable $c to the number of articles from author $aut (from the document "article.xml"). Those author elements for which the condition in the `where` (W) clause is true are selected, i.e., number of articles is above 100. The resulting bindings are sorted by the `order by` (O) clause on the author last

[4] http://www.w3.org/TR/xquery/

[5] http://www.w3.org/TR/xpath-full-text-10/

name. Finally, the `return` (R) clause creates for each binding $aut and $c in the result of the preceding clause a new author element that contains the last name element of the author, and the associated number of articles.

XQuery is a powerful XML query language. However, its text search capabilities are limited and, in addition, the result is a set of (new) XML fragments; no ranking is performed. Thus its usefulness in XML information retrieval is limited. This has led to the development of XQuery Full-Text [1].

XQuery Full-Text has been inspired by earlier query languages for searching structured text, e.g. ELIXIR [17], JuruXML [14], XIRQL [23]. The added text search capabilities come with the introduction of the *FTContainsExpr* expression, as shown in the following example:

```
//article[./title ftcontains {"XML", "retrieval"}
                                          all]//author
```

which returns the authors of articles whose title contains the words "XML" and "retrieval". XQuery Full-Text defines primitives for searching text, such as phrase, word order, word proximity, etc. For example, the following XQuery Full-Text expression:

```
//article[./title ftcontains  {"XML", "retrieval"}
     all window 6 words]//author
```

restricts the proximity of the matched words to appear within a window of six words in title elements.

To support the ranking of results, *FTScoreClause* expressions have been introduced to allow for the specification of score variables. For instance, the following query:

```
for $b score $s in //article[.//section ftcontains
                              {"XML", "retrieval"}  all]
      order by $s descending
      return <article title=``{$article/title}''
                              score=``{$s}''/>
```

iterates over all articles whose sections contain both "XML" and "retrieval" , where the `$b` variable binds the score of each such article to the score variable `$s`. These variables are used to return the titles of the articles and their scores in order of decreasing relevance.

XQuery Full-Text does not implement a specific scoring method, but it allows an implementation to proceed as it wishes. In other words, each XQuery Full-Text implementation can use a scoring method of its choice. Therefore, an appropriate implementation of XQuery Full-Text can allow ranking of results. From a user perspective, XQuery Full-Text may be viewed as far too complex, which is one of the reasons[6] the INEX community developed NEXI, a path-based query language with less expressiveness than a clause-based query language, as its query language. Nevertheless, XQuery Full-Text is needed in applications involving expert users, e.g. medical domain, patent industry, law.

[6] A second one was to keep the construction of the test collections manageable, for instance during the assessment task, see footnote 3.

3 Representation strategies

To retrieve documents relevant to a query, the first task of an information retrieval system is to index all documents forming the searched collection. The indexing task aims to obtain a representation of the content of documents (i.e. what each document is about), which can then be used to score each document according to how relevant it is to a given query. Classical indexing strategies in information retrieval make use of term statistics, the most common ones being the within-document term frequency, tf, and the inverse document frequency, idf. tf is the number of occurrences of a term in a document and reflects how well a term captures the topic of a document; a term that occurs frequently in a document can be considered a good description of the document content (apart from common words, referred to as stop words, e.g. "the", "every" in the English language). idf is the inverse number of documents in which a term appears and is used to reflect how well a term discriminates between relevant and non-relevant documents; a term that appears in all documents of the collection is not good at discriminating between the content of two documents, and hence their relevance or non-relevance.

With these term statistics, an index is build, for instance in the form of an inverted file, which gives for each term in the collection its idf, and for each document containing that term, the corresponding tf. Indexing algorithms for XML information retrieval require similar terms statistics, but at element level, i.e. they require so-called within-element term frequency, etf, and inverse element frequency, ief[7].

In XML information retrieval, there are no a priori fixed retrieval units. The whole document, a part of it (e.g. one of its section), or a part of a part (e.g. a paragraph in the section), that is, elements at all levels of granularity, all constitute potential answers to a given query. The simplest approach to allow the retrieval of elements at any level of granularity is to index all elements. Each element thus corresponds to a document, and conventional information retrieval representation techniques can be used. Term statistics (etf and ief) for each element are then calculated exactly in the same way as for tf and idf but based on the concatenation of the text of the element and that of its descendants (e.g. [61]).

This is the most common approach. It however raises an issue because of the nested nature of the units forming an XML document: the ief value of a term will consider both the element that contains that term and all elements that do so in virtue of being ancestors of that element. For instance, for a section element composed of two paragraph elements, the fact that a term appears in the paragraph implies that it also appears in the section. This "double" occurrence may have an adverse effect with respect to using ief to discriminate between relevant and non-relevant elements.

As a consequence, alternative means have been used to calculate ief. For instance, ief has been estimated across elements of the same type (e.g. [64]) or across documents (e.g. [18]). The former greatly reduces the impact of nested elements on the ief

[7]The indexing of a collection of documents involves other steps than calculating term statistics. These include tokenisation, stop word removal, stemming, etc [46]. In XML information retrieval, the same steps are applied, and other steps such as parsing the XML format, which are not discussed in this article. Also not discussed in this article is that an index of the structure is build in order to record the relationships between elements.

value of a term, but does not eliminate it if elements of the same type are nested within each other (as it is the case with the Wikipedia test collection used at INEX [21]). The latter is the same as using inverse document frequency, which completely eliminates nested elements. Experimental results reported in [56] indicate that estimating ief across documents shows slight improvement over using elements. However, other experimental results reported in [11] show that better performance was obtained estimating ief accross all elements than across elements of the same types. As of today, it is not yet clear what is the best way to estimate ief, whether the estimation strategy depends on the retrieval model and its artifacts used to rank elements, or whether the issue of nested elements actually matters. Further research is needed here.

An alternative to using the concatenated text in an element to estimate term statistics is to derive them through the aggregation of term statistics (both etf and ief) of the element's own text, and those of each of its children elements (e.g. [52, 31]). Aggregated-based ranking, discussed in Section 4.2.2, uses the aggregated representation of elements to rank elements.

A second alternative approach is to only index leaf elements[8]. This implies that term statistics will only be calculated for leaf elements, which can then be used to rank the leaf elements themselves. With such strategy, the ranking of non-leaf elements requires propagation mechanisms (discussed in Section 4.2.1) that combine the score of their children elements into that of the element [30]. Both this and the above (aggregation) strategies overcome the issue of nested elements with respect to the calculation of ief.

It has also been common to discard elements smaller than a given threshold (usually expressed in terms of number of words) [61], which are often considered not meaningful retrieval units (they are too small to make much sense as results). It was however argued in [59] that although the small elements should not be returned, they might still influence the scoring of enclosing elements, so they should still be indexed, in particular when a propagation mechanism for scoring non-leaf elements is used.

A final strategy [48, 18], referred to as selective indexing, is to only index those element types with the highest distribution of relevant elements in past relevance data. With this strategy, a separate index is built for each selected element type (e.g., for a collection of scientific articles, these types may include article, abstract, section, sub-section, paragraph). The statistics for each index are then calculated separately. Since each index is composed of terms contained in elements of the same type (and likely comparable size), more appropriate term statistics are generated. In addition, this approach greatly reduces the term statistics issue arising from nested elements, although it may not eliminate it. At retrieval time, the query is then ran in parallel on each index, and the list results (one for each index) are merged to provide a single list of results, as discussed in Section 4.2.3.

It is not yet clear which indexing strategy is the best, as obviously which approach to follow would depend on the collection, the types of elements (i.e., the DTD) and their relationships. In addition, the choice of the indexing strategy has an effect on the ranking strategy. An interesting research would be to investigate all indexing strategies

[8]A leaf element is one at the bottom of the document tree structure, i.e. an element with no children elements, or an element that is considered the smallest possible unit of retrieval.

within a uniform and controllable environment to determine those leading to the best performance, across, or depending, on the ranking strategies.

4 Ranking strategies

Given an indexed collection of XML documents, the next task of an XML information retrieval system is to return for each submitted query, with or without structural constraints, a list of XML elements ranked in order of their estimated relevance to that query. In information retrieval, retrieval models are used to calculate what is called a retrieval score (usually a value between 0 and 1), which is then used as a basis to rank documents. Many of the retrieval models developed for unstructured text (document) retrieval have been adapted to XML information retrieval to provide such a score at element level (Section 4.1). These scores may be used to directly generate the ranked list of elements, or as input to combination strategies required for some indexing strategies in order to rank elements at all levels of granularity (Section 4.2). For content-and-structure queries, in the context of INEX as expressed by the path-based query language NEXI (see Section 2.2), the structural constraints must be processed to provide results that not only satisfy the content, but also the structural criteria of such queries (Section 4.3). Finally, not all relevant elements should be returned as results, as they may contain overlapping content. This is because of the nested nature of XML documents, which often means that a parent and its child element may both be estimated as relevant, although to a different extent. Some processing is needed to deal with overlapping elements in result lists (Section 4.4).

4.1 Scoring strategies

Whatever the representation strategy, i.e. whether all elements or only a subset of them are indexed, a scoring function is used to estimate the relevance of these elements for a given query. With the propagation strategy (discussed in Section 4.2.1), the scoring function is applied to leaf elements only, whereas in other cases, it is applied to all potentially retrievable elements. Scoring functions used in XML information retrieval have been based on standard information retrieval models, such as the vector space, BM25, language models, to name a few. These have been adapted to incorporate XML-specific features. As an illustration, we describe a scoring function defined upon a language modeling framework inspired from [61].

Given a query $q = (t_1, t_2, ..., t_n)$ made of n terms t_i, given an element e and its corresponding element language model θ_e, the scoring function expressed by $P(e|q)$ is defined as follow:

$$P(e|q) \propto P(e)P(q|\theta_e)$$

$P(e)$ is the prior probability of relevance for element e and $P(q|\theta_e)$ is the probability of a query being generated by the element language model θ_e, and can be calculated as:

$$P(t_1, .., t_n|\theta_e) = \prod_{i=1}^{n} \lambda P(t_i|e) + (1 - \lambda)P(t_i|C)$$

$P(t_i|e)$ is the maximum likelihood estimate of term t_i in element e, $P(t_i|C)$ is the probability of query term t_i in the collection, and λ is the smoothing parameter. $P(t_i|e)$ is the element model based on element term frequency (modeling itf), whereas $P(t_i|C)$ is the collection model based on inverse element frequency (modeling ief).

One important XML feature is the length of an element. Indeed, it was shown in [35] that considering element length is necessary in XML information retrieval to cater for the wide range in element sizes. This can be incorporated by setting the prior probability $P(e)$ as follows:

$$P(e) = \frac{length(e)}{\sum_C length(e)}$$

where $length(e)$ is the length of element e. Examples of other XML-specific features used in XML information retrieval include the path length [34], the type of an element (its tag) [29], and the number of topics discussed in an element [5].

The size of elements forming XML documents varies greatly. For example, compare a paragraph to a section in a ten page scientific article. There are likely to be fewer terms indexing the paragraph than the section, leading to a higher chance of a vocabulary mismatch between a paragraph (or any small elements) and a query than between a section (or any large elements) and the same query. In addition, the fact that a paragraph element does not contain all query terms, but is contained in a section element that contains all query terms, is likely to be more relevant than if contained in a section element that does not contain all query terms.

More generally, the context of an element, i.e. the parent, all or some of its ancestors, or the entire document, can provide more evidence on what an element is or is not about. To incorporate the selected context(s) in estimating relevance, the score of an element is modified to include that of its (selected) context(s). The most common technique is to use the document containing the element as context (the document is also an element, albeit a large one, and corresponds to what is being referred to as the root element). This means combining the score of the element to that of the XML document containing that element, where the element and the document retrieval scores are estimated by an XML information retrieval model. The combination can be as simple as the average of the two scores [3]. A scaling factor can be used to emphasise the importance of one score compared to the other [48]. This technique (using element and document scores) has been shown to increase retrieval performance, in particular for long documents, and has been widely used in XML information retrieval.

4.2 Combination strategies

Three of the representation strategies described in Section 3 require combination strategies to provide a rank list of all potentially retrievable elements. These combination strategies are propagation (Section 4.2.1), aggregation (Section 4.2.2) and merging (Section 4.2.3).

4.2.1 Propagation

The propagation strategy is needed with the representation strategy that only indexes leaf elements. The relevance of the leaf elements for given queries is estimated on this indexing, resulting in retrieval scores for leaf elements. The relevance of non-leaf elements is estimated through a propagation mechanism, where the retrieval score of a non-leaf element is calculated on the basis of the retrieval scores of its descendant elements. The propagation starts from the leaf elements and moves upward in the document tree structure.

The most common propagation mechanism consists of a weighted sum of retrieval scores. For instance, the number of children elements of an element has been used as a weight [30]:

$$score(e, q) = D(m) \sum_{e_c} score(e_c, q)$$

where $score(., q)$ is the retrieval score of an element with respect to query q, e_c is a child element of e, m is the number of retrieved children elements of e, $D(m) = 0.49$ if m=1 (e has only one retrieved child element), and 0.99 otherwise. The value of $D(m)$, called the decay factor, depends on the number of retrieved children elements. If e has one retrieved child then the decay factor of 0.49 means that an element with only one retrieved child will be ranked lower than its child. If e has several retrieved children, the decay factor of 0.99 means that an element with many retrieved children will be ranked higher than its children elements. Thus, a section with a single relevant paragraph would be considered less relevant than the paragraph itself, as it is simply better to return the paragraph as returning the section does not add anything more. On the other hand, a section with several retrieved paragraphs will be ranked higher than any of the paragraphs, as it will allow users to access these several paragraphs through the returned section.

This approach, known as the GPX model, has been very successful within INEX, across test collections and retrieval scenarios. Another successful approach, implemented in the XFIRM system [59], is to define the weight used in the propagation based on the distance between an element and its retrieved leaf elements.

4.2.2 Aggregation

This combination strategy is applied when the representation of an XML element is defined as the aggregation of the representation of its own content (if any) and the representations of the content of its children elements (if any). Retrieval is then based on these aggregated representations. The representation of the element's own content is generated using standard indexing techniques, whereas an aggregation function is used to generate the representation of the non-leaf elements. The aggregation function can include parameters (referred to as e.g. augmentation factor [31]) specifying how the representation of an element is influenced by that of its children elements (a measure of the contribution of, for instance, a section to its embedding chapter). Aggregation is to be contrasted to propagation; in the former, the combination is applied to representations, whereas in the latter, it is applied to retrieval scores.

To illustrate aggregation, we describe anapproach based on the language modeling framework inspired from [52]. There, each element e is modeled by a language model $\theta_{e_{own}}$ based on its own content. Now assume that e has several children, e_j, each with their own language model θ_{e_j}. Let $P(t|\theta_{e_{own}})$ and $P(t|\theta_{e_j})$ be the probability of query term t being generated by the language models $\theta_{e_{own}}$ and θ_{e_j}, respectively. The language model, called θ_e, modeling the element e based on its own content and that of its children, is defined as a linear interpolation of language models:

$$P(t|\theta_e) = \lambda_{own} P(t|\theta_{e_{own}}) + \sum_{e_j} \lambda_j P(t|\theta_{e_j})$$

where

$$\lambda_{own} + \sum_{e_j} \lambda_j = 1$$

The λ parameters model the contribution of each language model (i.e., element) in the aggregation, here implemented as a linear combination. The ranking of the elements is then produced by estimating the probability that each element generates the query (e.g. similarly to the formulation described in Section 4.1). The effectiveness of the aggregation, however, depends heavily on the appropriate settings of the λ parameters, whose values are usually estimated through learning methods.

4.2.3 Merging

The last combination strategy is that of merging, which is needed when a selective indexing strategy is used. With this indexing strategy, a separate index is created for each selected type of elements (e.g. article, abstract, section, paragraph, etc). A query submitted to the XML information retrieval system is run against each index separately, resulting in separate ranked lists of e.g. article elements, section elements, paragraph elements, etc. These lists need to be merged to provide a single ranking, across all element types.

In [48], the vector space model is used to rank elements in each index. Let e be an element and q a query. The following scoring function is used:

$$score(e, q) = \frac{\sum_{t \in q} w(t, q) \times w(t, e) \times ief(t)}{||q|| \times ||e||}$$

where $w(.,.)$ is the term weight based on within-element (etf)/query term frequency, and $ief(.)$ is the inverse element frequency. To merge the lists, normalisation is performed to take into account the variation in size of the elements in the different indices (e.g. paragraph index vs article index). For each result list, the element scores are normalised with $score(q, q)$, which corresponds to the score of the query as if it was an element in the collection run against the corresponding index. This ensures that scores across indices are comparable. The lists are then merged based on the normalised scores.

4.3 Processing structural constraints

We described so far approaches that were developed and evaluated during the INEX campaigns to rank elements given the content condition of a query. Given a query consisting of terms, these approaches deliver a list of elements ranked according to how they have been estimated relevant to that query. As discussed in Section 2, content-and-structure query languages have been developed to allow users to specify structural constraints, e.g. "give me a section in an article about XML technology that also discusses in one of its section book search".

Within INEX, structural constraints are viewed as hints as to where to look to find relevant information. The reasons for this view are two-fold. First, it is well known that users of information retrieval systems do not always, or simply cannot, properly express the content criterion (i.e. select the most useful query terms) of their information need. It is very likely that this also holds for the structural criterion of the information need. For instance, a user asking for paragraph components on "XML retrieval evaluation measures" may not have realised that relevant content for this query is scattered across several paragraphs, all of them contained within a single section. For that user, it may make more sense to return the whole section instead of individual paragraphs. Second – and to some extent as a consequence of the first reason above – there is a strong belief in the XML information retrieval community that satisfying the content criterion is, in general, more important than satisfying the structural criterion. For instance, even if a user is looking for section components on a particular topic, returning to that user abstract components would still be satisfactory, as long as the content criterion is satisfied.

Two main approaches have been developed to process structural constraints in XML information retrieval following this so-called vague interpretation of the structural constraints in content-and-structure queries. A first approach is to build a dictionary of equivalent synonyms. If, for example, <p> corresponds to paragraph type and <p1> corresponds to the first paragraph in a sequence of paragraphs, it would be quite logical to consider <p> and <p1> as equivalent tags (e.g. [47, 58]). The dictionary can also be built from analysing past relevance data [50]. If in such a data set, for example, a query asked for <section> elements, then all types of elements assessed relevant for that query can be considered equivalent to the <section> tag. Thus with this approach, if the structural constraint refers to e.g. <section>, then any element that is of type considered equivalent to <section>, will satisfy that structural constraint.

A second technique is that of structure boosting. There, the retrieval score of an element is generated ignoring the structural constraint of the query, but is then boosted according to how the structural constraint is satisfied by the element. The element structure and the query structure are compared and a structure score is generated. This structure score can be based on comparing the paths (e.g. [64, 14], and/or the tags in the paths (e.g. [70]). An important issue here is to determine the appropriate level of boosting. i.e. how much the initial content-based score should be boosted by the structure score.

The above techniques and their variants are used to determine the relevance of an element according to the content condition and tag-based like structural constraints, e.g. "retrieve sections about XML retrieval". For more complex structural constraints,

as allowed by a path-based language such as NEXI, e.g. retrieve paragraphs about ranking algorithms contained in sections about XML retrieval", a first step is usually applied, which is to divide the query into two tag-based like sub-queries, e.g. "retrieve paragraphs about ranking algorithms" and "retrieve sections about XML retrieval". Each sub-query is then processed according to its content condition, and its tag-based like structural condition as described in the previous two paragraphs. Each sub-query results in a ranked list of elements. To generate a ranked list for the whole query, the two ranked lists are compared, e.g. only elements returned for the "paragraph" sub-query whose ancestors are contained among the elements returned for the "section" sub-query are then retrieved. The final score depends on the implementation of the contain operation, e.g. a simple set containment, or using fuzzy operators (e.g. [71]).

Techniques for processing structural constraints were evaluated in the context of INEX, where the relevance of an element was assessed based on content only. In other words, there was no assessment of whether, for instance, a section element was a better element type to return than another element type (if both were relevant according to their content). Also, considering the structural constraints did not usually increase retrieval performance, apart maybe at very early ranks [68]. This result may however be due to the evaluation methodology. More research is needed regarding the usefulness and the impact of structural constraints for XML information retrieval.

4.4 Removing overlaps

We recall that the aim of an XML information retrieval system is to return the most relevant elements for a given query. Because of the nested structure of XML documents, when an element has been estimated relevant to a given query (by any of the XML ranking strategies presented in this article), it is likely that its ancestors will also be estimated as relevant, although likely to a different extent. This is because the same text fragment can be contained in several elements along a same path (e.g. a paragraph, its enclosing sub-section, the enclosing section, etc). Thus the element itself, its ancestors and a number of its descendants may be contained in the result list, eventually leading to a considerable amount of redundant information being returned to users, which may not be acceptable to them [66].

The outcome of any of the ranking strategies described so far in Section 4 is a list of elements ranked according to their estimated relevance to a given query, without looking at the overlap issue. XML information retrieval systems may have to decide which elements should be returned from a list of relevant but overlapping elements. Several approaches have been proposed to generate overlap-free result lists. Their starting point usually consists of the list of elements returned as results to a query, which they then process.

The most common approach, referred to as brute-force filtering, selects the highest ranked element from the result list and removes any ancestor and descendent elements from lower ranks. The process is applied recursively. This approach relies on the provision of ranking strategies that rank, among overlapping elements, those that should be selected at higher ranks. However, the ranking may not be appropriate for the purpose of returning the list of the most relevant non-overlapping results. This has led to a number of alternative approaches, where the tree structure of the XML documents has

been considered to decide which elements to remove from a list of overlapping results.

In the first such approach [50], a notion of the usefulness of an element is introduced to decide which elements to remove. Usefulness is modeled through a utility function defined upon the retrieval score of an element, its size, and the amount of irrelevant information contained in its children elements (implemented as the "amount" of text contained in the non-retrieved children elements). An element with an estimated utility value higher than the sum of the utility values of its children is selected and its children are removed. Otherwise, the children elements whose utility values exceed some set threshold are selected and the element is removed.

An alternative approach [49] looks at the distribution of retrieved elements in the XML document's tree structure, in addition to their score, to select the elements to retain. For instance, an element that has many of its descendants retrieved, but which are evenly distributed in the corresponding sub-tree structure, and in addition has a similar score to the parent element, is selected. This is because already from that selected element, all its descendants, many of which are being estimated as relevant, can be accessed. Otherwise, its descendants are selected to be themselves further processed.

A third approach [55] calculates a new score for each element on the basis of the retrieval scores of its (if any) descendent elements. This is done through a bottom-up propagation mechanism, using for instance the maximum or average operation to recalculate the scores. These scores are used to generate a new ranked list, which is then filtered by selecting the highest ranked elements, and then removing either all ancestors or all descendants of that selected element from the list (e.g. brute-force filtering). The best performance were obtained using the maximum function and removing the descendants.

Techniques that explicitly considered the document logical (tree) structure to remove overlaps usually outperformed those that did not. There is, however, the issue of speed, as the removal of overlaps is done at query time, thus requiring efficient implementations. An interesting question would be to investigate the effect of the original result list (how good it is, and how we define "goodness") on the overlap removal strategy. There are indications that a good initial result list, where good depends on the definition of relevance in the context of XML information retrieval, leads to better overlap-free result list than a less good one [4].

5 Conclusion

XML information retrieval research is related to work on structured document retrieval. The term "structured document retrieval", which was introduced by the information retrieval community, refers to "passage retrieval" and "structured text retrieval". In passage retrieval, documents are first decomposed into passages (e.g. fixed-size text-windows of words [13], fixed discourses such as paragraphs [72], or topic segments through the application of a topic segmentation algorithm [33]). Passages are then retrieved as answers to a query (and have also been used to rank documents as answers to the query). Since 2007, INEX has a passage retrieval task [37].

Structured text retrieval is concerned with the development of models for querying and retrieving from structured text [7], where the structure is usually encoded with the

use of mark-up languages, such as SGML, and now predominantly XML. Examples of pre-XML/NEX approaches include [12, 44, 51]. Most of the early structured text retrieval models, however, do not return a ranked list of results. Approaches that were specifically developed for XML retrieval, but still pre-INEX include [60, 40, 16, 57]. A survey on indexing and searching XML retrieval is from [43], and two workshops on XML retrieval held at the SIGIR[9] conference were reported in [6, 15]. A recent overview on XML retrieval research is from [2].

Research on XML retrieval significantly flourished since the set-up of INEX, as the latter allowed the evaluation and comparison of XML information retrieval approaches. Nowadays, XML retrieval is almost a synonym for structured document retrieval, or structured text retrieval. In this article, we described many of the strategies used for representing and ranking XML elements, which were experimented on the INEX test collections. We also described query languages that were developed to access XML documents with respect to both content and structural conditions.

It is not yet possible to state which approaches, whether for querying, representing or ranking, or their combination, work best, since many factors are involved when deciding how relevant an element is to a given query (e.g. the size of an element, the type of element, the relevance of structurally related elements, the interpretation of the structural constraint, etc). Indeed, XML information retrieval can be regarded as a combination problem, where the challenge is to decide which evidence to combine and how to combine it for effective retrieval. We can however postulate that considering the context of the element, the size of the element, and the element own content (directly or using a propagation or aggregation strategy) to estimate that element relevance to a given query has been shown to be beneficial for XML information retrieval. An open research question is the processing of structural constraints, as so far, only limited improvement in retrieval performance has been observed with structure-and-content queries.

The querying, representation and ranking strategies described in this article have been developed with the purpose of estimating the relevance of an element for a given query, which is only one retrieval scenario. This may not necessarily be the end task in XML information retrieval. Returning overlap-free results is another retrieval scenario, one where users do not want redundant information (approaches developed for this purpose were described in Section 4.4). Another retrieval scenario investigated at INEX is the relevant in context task [45]. This task is concerned with returning the most relevant documents for a given query, and within each document, identifying the most relevant elements. Such a retrieval scenario was identified as important, if not expected, in a user study carried out within a software company regarding the benefit of focused retrieval [9].

In the relevant in context task, elements from the same documents are grouped together. A system could also create so-called "fictitious" documents, i.e. new documents made from some intelligent aggregation of elements coming from different documents, which is another retrieval scenario, currently receiving increasing attention in information retrieval research [41]. XQuery Full-Text would be an appropriate language for this task, as it allows the specification of new XML fragments to return as results.

[9] http://www.sigir.org/

Another retrieval scenario, also investigated at INEX, is the best in context task [45]. There, the aim is to identify the one and only best entry point in a document, i.e. the XML element, from where one could start reading relevant content. Such a retrieval scenario makes sense with a collection of relatively medium size documents (e.g. Wikipedia documents used at INEX since 2006 [21]).

Although not discussed in this article, two important issues in XML information retrieval are interface and interaction. Appropriate interfaces are needed to cater for the richer and likely more complex interaction between users and XML information retrieval systems, for example, with respect to expressing content-and-structure queries (e.g. [73]). Since 2004, INEX run an interactive track (iTrack) that looked at interaction issues in XML information retrieval [65]. One outcome of iTrack is that users did not like being returned (at least too much) redundant information (overlapping results). This led to the development of algorithms specifically dedicated to remove or reduce overlaps (see Section 4.4). A second outcome was that users expected to have not only access to relevant elements, but also the context of these elements (e.g. the document containing, or the parent element of, a retrieved element). This led to the proposal of a table of a content shown in conjunction to the element being accessed [62] or the use of heatmap highlighting relevant elements within a retrieved document [36].

Information retrieval approaches developed for querying, representing, or ranking are relevant to applications concerned with the effective access to repositories of documents annotated in XML, or similar mark-up languages. XML retrieval is becoming increasingly important in all areas of information retrieval, and in particular in the area of so-called focussed retrieval [67]. Current applications of XML information retrieval technologies already exist [54]. An example is that of book search [38], which is a research track being investigated at INEX since 2007 [39].

Acknowledgments

This article is based on two other articles on XML information retrieval co-written by the author, a book chapter on "Structured Text Retrieval" to appear in the second edition of [8], and an entry on "XML Retrieval" [42] to appear in the Encyclopedia of Database Systems [53]. The author would like to thank Benjamin Piwowarski and Anastasio Tombros for their comments on this article.

References

[1] AMER-Y.AHIA, S., BOTEV, C., DÖRRE, J., AND SHANMUGASUNDARAM, J. Full-Text extensions explained. *IBM Systems Journal 45*, 2 (2006), 335–352.

[2] AMER-YAHIA, S., AND LALMAS, M. XML search: languages, INEX and scoring. *SIGMOD Record 35*, 4 (2006), 16–23.

[3] ARVOLA, P., JUNKKARI, M., AND KEKÄLÄINEN, J. Generalized contextualization method for XML information retrieval. In *ACM CIKM International Conference on Information and Knowledge Management, Bremen, Germany* (2005), pp. 20–27.

[4] ASHOORI, E. *Using Topic shifts in Content-oriented XML Retrieval*. PhD thesis, Queen Mary, University of London, 2009.

[5] ASHOORI, E., LALMAS, M., AND TSIKRIKA, T. Examining topic shifts in content-oriented XML retrieval. *International Journal on Digital Libraries 8*, 1 (2007), 39–60.

[6] BAEZA-YATES, R., FUHR, N., AND MAAREK, Y. Second edition of the "XML and information retrieval" workshop held at SIGIR'2002, Tampere, Finland. *SIGIR Forum 36*, 2 (2002), 53–57.

[7] BAEZA-YATES, R. A., AND NAVARRO, G. Integrating contents and structure in text retrieval. *SIGMOD Record 25*, 1 (1996), 67–79.

[8] BAEZA-YATES, R. A., AND RIBEIRO-NETO, B. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[9] BETSI, S., LALMAS, M., TOMBROS, A., AND TSIKRIKA, T. User expectations from XML element retrieval. In *29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA* (2006), pp. 611–612.

[10] BLANKEN, H., GRABS, T., SCHEK, H.-J., SCHENKEL, R., AND WEIKUM, G., Eds. *Intelligent Search on XML Data, Applications, Languages, Models, Implementations, and Benchmarks* (2003), vol. 2818, Springer.

[11] BROSCHART, A., SCHENKEL, R., THEOBALD, M., AND WEIKUM, G. TopX @ INEX 2007. In *Focused access to XML documents, 6th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2007, Dagstuhl Castle, Germany, Selected Papers* (2008).

[12] BURKOWSKI, F. Retrieval activities in a database consisting of heterogeneous collections of structured text. In *15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Copenhagen, Denmark* (1992), pp. 112–125.

[13] CALLAN, J. Passage-level evidence in document retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval* (1994), Springer-Verlag New York, Inc., pp. 302–310.

[14] CARMEL, D., MAAREK, Y., MANDELBROD, M., Y.MASS, AND SOFFER, A. Searching XML documents via XML fragments. In *26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Toronto, Canada* (2003), pp. 151–158.

[15] CARMEL, D., MAAREK, Y. S., AND SOFFER, A. XML and information retrieval: a sigir 2000 workshop. *SIGIR Forum 34*, 1 (2000), 31–36.

[16] CHIARAMELLA, Y., MULHEM, P., AND FOUREL, F. A model for multimedia information retrieval. Tech. rep., University of Glasgow, 1996.

[17] CHINENYANGA, T. T., AND KUSHMERICK, N. Expressive Retrieval from XML Documents. In *24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New Orleans, Louisiana* (2001), pp. 163–171.

[18] CLARKE, C. Controlling overlap in content-oriented XML retrieval. In *28th annual international ACM SIGIR conference on Research and development in information retrieval, Salvador, Brazil* (2005), pp. 314–321.

[19] CLARKE, C. A., CORMACK, G., AND BURKOWSKI, F. An algebra for structured text search and a framework for its implementation. *Comput. J. 38*, 1 (1995), 43–56.

[20] COHEN, S., MAMOU, J., KANZA, Y., AND SAGIV, Y. XSEarch: A Semantic Search Engine for XML. In *29th International Conference on Very Large Data Bases, Berlin, Germany* (2003), pp. 45–56.

[21] DENOYER, L., AND GALLINARI, P. The Wikipedia XML Corpus. *SIGIR Forum 40*, 1 (2006), 64–69.

[22] FUHR, N., GÖVERT, N., KAZAI, G., AND LALMAS, M., Eds. *INitiative for the Evaluation of XML Retrieval (INEX). Proceedings of the First INEX Workshop. Dagstuhl, Germany, December 8–11, 2002* (Sophia Antipolis, France, 2003), ERCIM Workshop Proceedings, ERCIM.

[23] FUHR, N., AND GROSSJOHANN, K. XIRQL: An XML query language based on information retrieval concepts. *ACM Transaction on Information Systems 22*, 2 (2004), 313–356.

[24] FUHR, N., KAMPS, J., LALMAS, M. ., MALIK, S., AND TROTMAN, A., Eds. *Focused Access to XML Documents, 6th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2007, Dagstuhl Castle, Germany, December 17-19, 2007. Selected Papers* (2008).

[25] FUHR, N., LALMAS, M., AND MALIK, S., Eds. *INitiative for the Evaluation of XML Retrieval (INEX). Proceedings of the Second INEX Workshop. Dagstuhl, Germany, December 15–17, 2003* (2004).

[26] FUHR, N., LALMAS, M., MALIK, S., AND KAZAI, G., Eds. *Advances in XML Information Retrieval and Evaluation: Fourth Workshop of the INitiative for the Evaluation of XML Retrieval (INEX 2005)* (2006), vol. 3977 of *Lecture Notes in Computer Science*, Springer-Verlag.

[27] FUHR, N., LALMAS, M., MALIK, S., AND SZLÁVIK, Z., Eds. *Advances in XML Information Retrieval, Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl Castle, Germany, December 6-8, 2004, Revised Selected Papers* (2005), vol. 3493 of *Lecture Notes in Computer Science*, Springer.

[28] FUHR, N., LALMAS, M., AND TROTMAN, A., Eds. *Comparative Evaluation of XML Information Retrieval Systems, 5th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2006* (2007), vol. 4518 of *Lecture Notes in Computer Science*, Springer-Verlag.

[29] GERY, M., LARGERON, C., AND THOLLARD, F. Probabilistic document model integrating XML structure. In *INEX 2007 Pre-proceedings* (2007), pp. 139–149.

[30] GEVA, S. GPX - Gardens Point XML IR at INEX 2005. In *Advances in XML Information Retrieval and Evaluation, 4th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2005, Dagstuhl Castle, Germany, Revised Selected Papers* (2006), pp. 240–253.

[31] GÖVERT, N., ABOLHASSANI, M., N.FUHR, AND GROSSJOHANN, K. Content-oriented XML retrieval with HyRex. In *First Workshop of the INitiative for the Evaluation of XML Retrieval (INEX), Schloss Dagstuhl, Germany* (2002), pp. 26–32.

[32] GÖVERT, N., AND KAZAI, G. Overview of the INitiative for the Evaluation of XML retrieval (INEX) 2002. In *First Workshop of the INitiative for the Evaluation of XML Retrieval (INEX), Schloss Dagstuhl, Germany* (2002), pp. 1–17.

[33] HEARST, M. TextTiling: Segmenting Text into Multi-Paragraph Subtopic Passages. *Computational Linguistics 23*, 1 (1997), 33–64.

[34] HUANG, F., WATT, S., HARPER, D., AND CLARK, M. Compact representations in XML retrieval. In *Comparative Evaluation of XML Information Retrieval Systems, 5th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2006, Dagstuhl Castle, Germany, Revised and Selected Papers* (2006), pp. 64–72.

[35] KAMPS, J., DE RIJKE, M., AND SIGURBJÖRNSSON, B. Length normalization in XML retrieval. In *27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Sheffield, UK* (2004), pp. 80–87.

[36] KAMPS, J., KOOLEN, M., AND SIGURBJÖRNSSON, B. Filtering and Clustering XML Retrieval Results. In *Comparative Evaluation of XML Information Retrieval Systems, 5th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2006, Dagstuhl Castle, Germany, Revised and Selected Papers* (2006), pp. 121–136.

[37] KAMPS, J., PEHCEVSKI, J., KAZAI, G., LALMAS, M., AND ROBERTSON, S. INEX 2007 Evaluation Metrics. In *Focused access to XML documents, 6th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2007, Dagstuhl Castle, Germany, Selected Papers* (2008).

[38] KANTOR, P. B., KAZAI, G., MILIC-FRAYLING, N., AND WILKINSON, R., Eds. *Proceedings of the 2008 ACM Workshop on Research Advances in Large Digital Book Repositories, BooksOnline 2008, Napa Valley, California, USA, October 30, 2008* (2008), ACM.

[39] KAZAI, G., AND DOUCET, A. Overview of the INEX 2007 Book Search Track (BookSearch '07). *SIGIR Forum 42*, 1 (2008), 2–15.

[40] LALMAS, M. Dempster-shafer's theory of evidence applied to structured documents: Modelling uncertainty. In *20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Philadelphia, PA, USA* (1997), pp. 110–118.

[41] LALMAS, M., AND MURDOCK, V., Eds. *ACM SIGIR Workshop on Aggregated Search* (Singapore, 2008).

[42] LALMAS, M., AND TROTMAN, A. XML Retrieval. In *Encyclopedia of Database Systems*. Springer, 2009. To Appear.

[43] LUK, R. P., LEONG, H. V., DILLON, T., CHAN, A. S., CROFT, W. B., AND ALLAN, J. A survey in indexing and searching XML documents. *Journal of the American Society for Information Science and Technology 53*, 6 (2002), 415–437.

[44] MACLEOD, I. Storage and retrieval of structured documents. *Information Processing & Management 26*, 2 (1990), 197–208.

[45] MALIK, S., TROTMAN, A., LALMAS, M., AND FUHR, N. Overview of INEX 2006. In *Comparative Evaluation of XML Information Retrieval Systems, 5th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2006, Dagstuhl Castle, Germany, December 17-20, 2006, Revised and Selected Papers* (2007), pp. 1–11.

[46] MANNING, C., RAGHAVAN, P., AND SCHUTZE, H., Eds. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[47] MASS, Y., AND MANDELBROD, M. Retrieving the most relevant XML Components. In *INEX 2003 Proceedings* (2003), pp. 53–58.

[48] MASS, Y., AND MANDELBROD, M. Component Ranking and Automatic Query Refinement for XML Retrieval. In *Advances in XML Information Retrieval, Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl Castle, Germany, Revised Selected Papers* (2005), pp. 73–84.

[49] MASS, Y., AND MANDELBROD, M. Using the INEX Environment as a Test Bed for Various User Models for XML Retrieval. In *Advances in XML Information Retrieval and Evaluation, 4th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2005, Dagstuhl Castle, Germany, Revised Selected Papers.* (2006), pp. 187–195.

[50] MIHAJLOVIC, V., RAMÍREZ, G., WESTERVELD, T., HIEMSTRA, D., BLOK, H. E., AND DE VRIES, A. TIJAH Scratches INEX 2005: Vague Element Selection, Image Search, Overlap, and Relevance Feedback. In *Advances in XML Information Retrieval and Evaluation, 4th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2005, Dagstuhl Castle, Germany, Revised Selected Papers* (2006), pp. 72–87.

[51] NAVARRO, G., AND BAEZA-YATES, R. Proximal nodes: A model to query document databases by content and structure. *ACM Transactions on Information Systems 15*, 4 (1997), 400–435.

[52] OGILVIE, P., AND CALLAN, J. Hierarchical language models for XML component retrieval. In *Advances in XML Information Retrieval, Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl Castle, Germany, Revised Selected Papers* (2005), pp. 224–237.

[53] OZSU, M., AND LIU, L., Eds. *Encyclopedia of Database Systems*. Springer, 2009. To Appear.

[54] PHARO, N., AND TROTMAN, A. The use case track at INEX 2006. *SIGIR Forum 41*, 1 (2007), 64–66.

[55] POPOVICI, E., MÉNIER, G., AND MARTEAU, P.-F. SIRIUS XML IR System at INEX 2006: Approximate Matching of Structure and Textual Content. In *Comparative Evaluation of XML Information Retrieval Systems, 5th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2006, Dagstuhl Castle, Germany, Revised and Selected Papers* (2007), pp. 185–199.

[56] RAMÍREZ, G. *Structural Features in XML Retrieval*. PhD thesis, University of Amsterdam, 2007.

[57] RÖLLEKE, T., LALMAS, M., KAZAI, G., RUTHVEN, I., AND QUICKER, S. The accessibility dimension for structured document retrieval. In *Advances in Information Retrieval, 24th BCS-IRSG European Colloquium on IR Research, Glasgow, UK* (2002), pp. 284–302.

[58] SAUVAGNAT, K., BOUGHANEM, M., AND CHRISMENT, C. Answering content and structure-based queries on XML documents using relevance propagation. *Information Systems 31*, 7 (2006), 621–635.

[59] SAUVAGNAT, K., HLAOUA, L., AND BOUGHANEM, M. XFIRM at INEX 2005: Ad-Hoc and Relevance Feedback Tracks. In *Advances in XML Information Retrieval and Evaluation, 4th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2005, Dagstuhl Castle, Germany, Revised Selected Papers* (2006), pp. 88–103.

[60] SCHLIEDER, T., AND MEUSS, M. Result ranking for structured queries against xml documents. In *In DELOS Workshop: Information Seeking, Searching and Querying in Digital Libraries, Zurich, Switzerland* (2000).

[61] SIGURBJORNSSON, B., KAMPS, J., AND DE RIJKE, M. An element-based approch to XML retrieval. In *Proceedings INEX 2003 Workshop* (2004), pp. 19–26.

[62] SZLÁVIK, Z., TOMBROS, A., AND LALMAS, M. Feature- and query-based table of contents generation for xml documents. In *Advances in Information Retrieval, 29th European Conference on IR Research, ECIR 2007, Rome, Italy, April 2-5, 2007, Proceedings* (2007), pp. 456–467.

[63] THEOBALD, A., AND WEIKUM, G. The Index-Based XXL Search Engine for Querying XML Data with Relevance Ranking. In *EDBT* (2002), pp. 477–495.

[64] THEOBALD, M., SCHENKEL, R., AND WEIKUM, G. TopX and XXL at INEX 2005. In *Advances in XML Information Retrieval and Evaluation, 4th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2005, Dagstuhl Castle, Germany, Revised Selected Papers* (2006), pp. 282–295.

[65] TOMBROS, A., LARSEN, B., AND MALIK, S. The interactive track at INEX 2004. In *Advances in XML Information Retrieval, Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl Castle, Germany, Revised Selected Papers* (2005), pp. 410–423.

[66] TOMBROS, A., MALIK, S., AND LARSEN, B. Report on the INEX 2004 interactive track. *SIGIR Forum 39*, 1 (2005), 43–49.

[67] TROTMAN, A., GEVA, S., AND KAMPS, J. Report on the SIGIR 2007 workshop on focused retrieval. *SIGIR Forum 41*, 2 (2007), 97–103.

[68] TROTMAN, A., AND LALMAS, M. Why structural hints in queries do not help XML retrieval. In *29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA* (2006), pp. 711–712.

[69] TROTMAN, A., AND SIGURBJORNSSON, B. Narrowed Extended XPath I (NEXI). In *Advances in XML Information Retrieval, Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl Castle, Germany, Revised Selected Papers* (2005), pp. 16–40.

[70] VAN ZWOL, R. B^3-SDR and Effective Use of Structural Hints. In *Advances in XML Information Retrieval and Evaluation, 4th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2005, Dagstuhl Castle, Germany, Revised Selected Papers* (2006), pp. 146–160.

[71] VITTAUT, J.-N., PIWOWARSKI, B., AND GALLINARI, P. An algebra for structured queries in bayesian networks. In *Advances in XML Information Retrieval, Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl Castle, Germany, December 6-8, 2004, Revised Selected Papers* (2004), pp. 100–112.

[72] WILKINSON, R. Effective retrieval of structured documents. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval* (1994), Springer-Verlag New York, Inc., pp. 311–317.

[73] ZWOL, R., BAAS, J., VAN OOSTENDORP, H., AND WIERING, F. Bricks: The Building Blocks to Tackle Query Formulation in Structured Document Retrieval. In *Advances in Information Retrieval, 28th European Conference on IR Research, ECIR 2006, London* (2006), pp. 314–325.