XML Search: Languages, INEX and Scoring

Sihem Amer-Yahia
AT&T Labs Research
180 Park Ave.
Florham Park, NJ
sihem@research.att.com

Mounia Lalmas
Queen Mary, University of London
Mile End Road
London E1 4NS
mounia@dcs.qmul.ac.uk

1 Introduction

The development of approaches to access XML content has generated a wealth of issues in information retrieval (IR) and database (DB) (e.g., [2, 15, 17, 20, 19, 47, 26, 32, 24]). While the IR community has traditionally focused on searching unstructured content, and has developed various techniques for ranking query results and evaluating their effectiveness, the DB community has focused on developing query languages and efficient evaluation algorithms for highly structured content. Recent trends in DB and IR research demonstrate a growing interest in merging IR and DB techniques for accessing XML content. Support for a combination of "structured" and full-text search for effectively querying XML documents was unanimous in a recent panel at SIGMOD 2005 [3], and is being widely studied in the IR community [20].

This paper presents an overview of the recent research in XML search, and is composed of 5 sections. Section 2 describes query languages for XML search with an emphasis on the role of document structure in their design and semantics. These languages are mainly inspired from DB work. In IR, many proposals for accessing XML content have been made possible due to INEX which has allowed the evaluation and the comparison of the effectiveness of different XML search approaches. Section 3 gives an overview of INEX, the INitiative for the Evaluation of XML retrieval (INEX) [20]. In Section 4, we describe approaches that have been developed in order to score XML elements according to their relevance. Much of these works were developed and investigated within INEX, but the section also includes work that has been done in DB. Section 5 concludes with a brief summary and discussion of open issues in XML search.

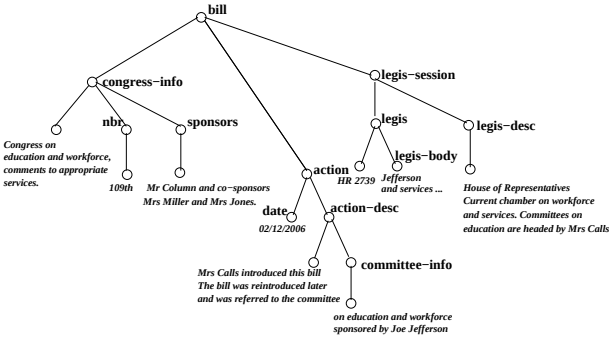

Figure 1: Document Excerpt from the LoC

This paper is based on two recent tutorials on XML search [7, 8] and is by no means exhaustive. It is meant as a reference overview for query languages, INEX, and scoring methods for XML, with a strong focus on the latter two. Readers are invited to explore the intricacies of each language, scoring method and INEX in the appropriate references at the end of this paper.

2 Query Languages

Many query languages with varying expressiveness [12] have been proposed for XML search. These languages range from simple keyword search to sophisticated proximity and stemming conditions on keywords combined with complex queries on structure. Document structure plays a major role in the design and semantics of these languages. Even when the language supports Boolean

search only, document fragments may be returned, as opposed to whole documents.

We use the example document in Figure 1, extracted from the Library of Congress (LoC) collection [31]. The document describes a bill on education and workforce and exhibits a mix of structured and text information.

In XML search, document structure is used to determine, on behalf of users, which document fragments are more meaningful to return as query answers. It is also used to specify query conditions on structure that limit the search context to specific XML elements, as opposed to whole documents. We classify existing languages for XML search according to their use of conditions on document structure in the query.

Keyword-Only Queries. Languages in this family are a natural evolution from traditional IR languages and are expressed as a conjunction of keywords. The main characteristic of these languages is their ability to return XML fragments. Examples of such languages are found in XRANK [24], nearest-neighbor queries [42], XKSEARCH [55], and NEXI content-only queries [48] developed by INEX [20] (See Section 3). The most notorious method in DB, is to compute lowest common ancestors (LCAs) of query keywords. For example, the query `''Jefferson'' && ''workforce''` applied to the document in Figure 1, would return `<committee-info>` and `<legis-session>` elements as query answers. Variants of the LCAs include meaningful LCAs as in [30] and in XIRQL [19], which are pre-determined XML fragments. Other methods, used in IR, are described in Section 4.

Tag and Keyword Queries. This family of languages includes XSEarch [17] which allows to annotate keywords in the query with tags. For example, the query above could be enhanced as follows `committee-info:Jefferson workforce`, in which case only `<committee-info>` will be returned. XSEarch may also consider semantically related tags as query results.

Path and Keyword Queries. This family of languages exhibits more sophisticated conditions on structure ala XPath [53]. It includes XPath 2.0 [53], XIRQL [19], XXL [47], and NEXI Content-And-Structure (CAS) queries [48] (See Section 3). While the exact syntax of each language is not of interest here, they all agree on an existential interpretation of the path conditions.

XQuery and Keyword Queries. Languages in this family combine the full power of XQuery [52] and a range of *full-text* predicates (as opposed to simple Boolean predicates). Schema-free XQuery offers the ability to specify LCAs with XQuery. For example, the query below returns a bill if there exists a meaningful LCA of `<legis>` and `<legis-desc>` elements, in this case `<legis-session>`, containing the keywords `''Jefferson''` and `''workforce''` respectively. A meaningful LCA is defined as an LCA which makes sense in the context of a given application. For example, returning footnote or paragraph elements may be too fine a granularity to the user.

```
for $b in //bills/bill
    $tmp1 in $b//legis
    $tmp2 in $b//legis-desc
where fn:contains($tmp1, 'Jefferson')
 and  fn:contains($tmp2, 'workforce')
      and exists mlcas($tmp1, $tmp2)
return <result> {$b} </result>
```

TeXQuery [2] and its W3C successor, XQuery Full-Text [54], extend XQuery with composable full-text search predicates such as proximity distance and stemming. The query below written in XQuery Full-Text, returns newly constructed elements containing the number and legislation description of bills produced after January 12th, 2006 and which contain `''Jefferson''` and `''workforce''` within a window of 5 words and using stemming on `''workforce''` if necessary.

```
for $b in //bills/bill
where $b ftcontains 'Jefferson' &&
       'workforce' with stemming
       window 5 words and
       $b//date > 01/12/2006
return <res> {$b//nbr,
              $b//legis-desc} </res>
```

3 INEX Evaluation

There has been many XML search systems that have been developed to implement the query languages described in Section 2, many of which being evaluated within the INEX campaign. In this section, we describe how INEX is providing methods to evaluate the effectiveness of query results.

Evaluating how effective XML search systems are, requires test-beds where the evaluation paradigms are provided according to criteria that take into account the imposed structural aspects. In March 2002, the INitiative for the Evaluation of XML retrieval (INEX) [20] started to address these issues. The aim of the INEX initiative is to establish an infrastructure and to provide means, in the form of a large test collection and appropriate scoring methods, for evaluating how effective are content-oriented XML search systems (as opposed to structure-oriented systems).

Evaluating retrieval effectiveness is typically done by using test collections assembled specifically for evaluating particular retrieval tasks. A test collection consists of a set of documents, a set of user requests (the so-called topics, or queries) and relevance assessments of the documents with respect to the queries. Most of existing test collections treat documents as atomic units and make assumptions that become invalid in the context of XML retrieval, which is mainly due to the lack of a predefined uniform unit of retrieval and the dependency that exists among retrievable components. As such, the characteristics of traditional test collections have been adjusted in order to appropriately evaluate content-oriented XML retrieval effectiveness.

Document Collection The INEX corpus is composed of the full-texts, marked up in XML, of 16.819 articles of the IEEE Computer Society's publications from magazines and transactions, covering the period of 1995-2004, and totaling 735 megabytes in size. On average an article contains 1,532 XML nodes, where the average depth of a node is 6.9. The overall structure of a typical article consists of a frontmatter (containing e.g. title, author, publication information and abstract), a body (consisting of e.g. sections, sub-sections, sub-sub-sections, paragraphs, tables, figures, lists, citations) and a backmatter (including bibliography and author information).

Topics INEX defined two types of topics: *Content-only (CO)* queries which are similar to those used in TREC and, *Content-and-structure (CAS)* queries which contain structural constraints that can refer to where to look for the relevant elements (i.e. support elements) and what types of elements to retrieve (i.e. target elements). NEXI [48] is the query language defined for this purpose and is based on XPath [53]. It should be noted that NEXI is more focused on querying content (content-oriented retrieval) than many of the languages presented in Section 2. As in TREC, an INEX topic consists of the standard title, description and narrative fields. Up to date, the INEX has accumulated a total of 268 topics, most of them with associated relevance assessments.

Tasks The main INEX activity is the ad-hoc retrieval task. Ad-hoc retrieval in IR is described as a simulation of how a library might be used, and it involves the searching of a static set of documents using a new set of topics. Two ad hoc retrieval sub-tasks have been identified since 2002, depending on how structural constraints are expressed. In the CO sub-task, using CO queries, it is left to the retrieval system to identify the most appropriate XML elements to return to the user. Depending on how we assume a user wants the output of an XML retrieval system to be, three different strategies have been defined in 2005. In a focused strategy, we assume that a user prefers a single element that most exhaustively discusses the topic of the query, while at the same time it is most specific to that topic. In a thorough strategy, we assume that a user prefers all highly exhaustive and specific elements, and in a fetch and browse strategy we assume that a user is interested in highly exhaustive and specific elements that are contained only within highly relevant articles. In the CAS sub-task, using CAS topics, structural constraints can be interpreted as strict or vague (they are viewed as hints). Strict and vague interpretation can be applied to both support and target elements, giving a total of four strategies for this sub-task.

Relevance XML elements forming a document can be nested. Some elements are large (e.g., sections) and others small (e.g., paragraphs). Since retrieved elements can be

at any level of granularity, an element (the larger element) and one of its child elements (the smaller element) can both be relevant to a given query, but the child element may be more focused to that given query than its parent element and may thus be a better element to retrieve [38].

INEX uses two graded dimensions to express relevance: exhaustivity and specificity. Exhaustivity is defined as a measure of how exhaustively an XML element discusses the topic of request, while specificity is defined as a measure of how focused the element is on the topic of request (i.e. discusses no other, irrelevant topics). Exhaustivity refers to the standard relevance criterion used in IR, whereas specificity provides a measure with respect to the size of a component as it measures the ratio of relevant to non-relevant content within an element. The combination of the two dimensions is used to identify those relevant XML elements, which are both exhaustive and specific to the topic of request and hence represent the most appropriate unit of information to return to the user.

Metrics Since its launch in 2002, INEX has been challenged by the issue of how to measure an XML information access system's effectiveness. Up to 2004, INEX used a metric based on the measure of precall, which computes the probability that an XML element viewed by the user is relevant. That is, it interprets precision as the probability that an XML element viewed by a user is relevant at standard recall values. To apply the metric, the two relevance dimensions are mapped to a single relevance scale using quantisation functions. For example, a strict quantisation function is used to evaluate retrieval methods with respect to their capability of retrieving highly relevant elements. A generalised function is used to credit retrieved elements according to their degree of relevance, thus also allowing to reward less relevant elements.

The latter is important as it allows considering near-misses when calculating effectiveness performance. Due to the lack of an atomic predefined unit of retrieval as well as the increased richness of the user's interaction with the system (i.e., browsing), users have access to other, structurally related components from a returned result element. Near-misses are elements, which may be themselves not exactly relevant to the user's query, but from where users can access relevant content. The idea is that XML retrieval approaches are partially rewarded for finding such elements, as it is still better to return near-misses than irrelevant elements.

The metric, in its current version, cannot consider overlapping results. This is because, in INEX, the recall-base (the set of relevant elements for each given query) consists of a large proportion of overlapping elements (if an element is relevant, so is its parent element). This so-called overpopulated recall-base can lead to misleading effectiveness results because the recall-base contains more relevant elements than an ideal system should in fact retrieve. In fact, perfect recall can only be reached by systems that return all the relevant elements of the recall-base, *including* all the overlapping elements [29].

Therefore, INEX adopted in 2005 the eXtended Cumulated Gain metric, XCG [28], which considers both system and user-oriented evaluation aspects. User-oriented measures allow to reason about a system's ability to satisfy users, and typically focus on the early ranks of a system's output as users are more likely to limit their search to these results. System-oriented measures allow system developers to obtain an overall picture of performance.

4 Scoring

XML search systems aim at providing more precise access to XML documents by retrieving document components (the so-called XML elements) instead of whole documents in response to users' queries. This was illustrated in the description of INEX and non-INEX XML search languages in Section 2. In those languages, XML elements of any granularity (e.g., a paragraph or the section enclosing it) are potential answers to a query, as long as they are relevant. This constitutes a major departure from traditional IR: XML retrieval systems need not only score elements with respect to their relevance to a query, but also determine the appropriate level of component granularity to return to users. In this section, we discuss the main challenges associated with scoring and returning elements at the right level of granularity. We also briefly illustrate how these challenges have been or are currently being addressed in INEX.

Term and element statistics Classical scoring methods in IR make use of term and document statistics, the most common ones being term frequency, tf, and document

frequency df. tf is the number of occurrences of a term in a document and df is the number of documents in which the term appears. Scoring at element level requires statistics at element level. It is not straightforward to simply replace document by element (etf and ef), since elements in XML documents are nested. Suppose that a section element is composed of two paragraph elements. How should we compute the etf and ef values of a term in the paragraphs and the section?

One approach in [33] is to build an index for each element type (e.g., article, section, paragraph, etc), and compute the statistics for each element type separately. A query is then ran in parallel on each index (e.g., using any or several scoring models). The scores in each result set are normalized, so that scores from different indices are comparable. Finally, results are merged into a single result set. This avoids problems arising from nested elements as elements of different types are treated separately. A different approach is to compute etf and ef statistics for leaf-elements only, which are then used to score the leaf-elements themselves. All non-leaf elements are scored based on some (weighted) combination of the score of their children elements. The propagation of score starts from the leaf-elements and can consider the distance between the element being considered and its descendant leaf-elements [22, 40].

Many approaches ignore element nesting by simply calculating ef with respect to all elements of the collection [43], or partly consider it by estimating ef across elements of the same type [45] or using df [16]. etf is often calculated based on the concatenation of the text of the element and that of its descendants [36, 45].

It is not yet clear what works best, as obviously which approach to follow would depend on the collection, the types of elements (i.e., the DTD) and their relationships, and the scoring method. An interesting research would be to investigate all developed term and element statistics approaches within a uniform and controllable environment to determine those leading to the best performance.

Obtaining consistent and meaningful statistics for effective retrieval can be viewed as the more general problem of the *combination of evidence*. In XML retrieval, term and element statistics are not enough; element size has been shown to be important [44, 35]. The next paragraphs describe other types of evidence.

Structure Statistics Not all elements are equally likely to be appreciated as satisfactory answers to an information need. The structure in XML documents should be used to decide which elements are likely to trigger higher user satisfaction. This is usually done by computing some statistics on the structure itself. One approach [33, 16] is to only index and/or retrieve those element types with the highest distribution of relevant elements in past relevance assessment sets. In INEX, selected types included section, abstract, sub-section, paragraph element type. A different approach [43] is to index only those element types whose average length was above a given threshold. A related strategy is to discard at retrieval all elements smaller than a given threshold (usually expressed in terms of number of words). Both strategies strategy will indeed remove elements of a particular type, which are often considered to be too small to act as a meaningful retrieval unit. [40] however recently showed, that although the so-called small elements should not be returned, they might still influence the scoring of enclosing elements, so they should still be indexed.

Relationships Statistics XML documents are not just documents that are made of components of various types. There is a relationship between the elements, as provided by the logical structure of the XML mark-up. An element, unless it is a root element, has a parent element, which itself may have a parent element. Similarly, non-leaf elements have children elements, and so on. It makes sense to exploit element relationships to score elements.

Some approaches, like those based on the Bayesian network or hierarchical language modeling [50, 36] explicitly capture element relationships. They can associate, explicitly or implicitly, statistics to structural relationships through learning to capture the strength of those relationships. These models need extensive training, which is computationally expensive. A simplified hierarchical language model that reduces the parameter estimation complexity was proposed in [37].

Other approaches propagate terms statistics along the structure, where it is then possible to weight the propagation depending on the relationships [22, 40]. The weighting parameters are usually empirically determined.

A particular relationship is that of the element and its root element. For instance, in INEX, the root element

of any element is the article element. Considering this particular relationship has often shown to improve performance (e.g. pivot in [34], root-based contextutalitaion [9], article level language model [43]).

Overlapping Elements It is one task to provide a score expressing how relevant an element is to a query and a different task to decide, from several overlapping relevant elements, which one to return as the *best* answer. For instance, returning a paragraph and its enclosing section should be avoided to ensure that users do not get to see the same information several times. Deciding which element to return obviously depends on the application and its user models. For instance, in INEX, the best element is one that is highly relevant, but also specific to the topic (i.e., does not discuss unrelated topic).

Three types of approaches have been applied within INEX to remove overlap. A common one is to select the highest scored element from each path and return that element [40, 22]. Additional filtering can be applied such as discarding all elements of a given type (e.g., article elements even if they are highly scored [50]). Although the tree structure of a document may have been considered at indexing and/or retrieval time, it is ignored when removing overlap, as elements are treated independently.

An approach that does not treat elements independently when removing overlap is [16], where elements are re-ranked by iteratively adjusting the score of those elements contained in or containing higher ranked elements in the result list. Overlap is controlled by re-weighting terms occurring in the higher ranked elements to reflect their reduced importance. A second approach [34] looks at the distribution of relevant elements in the tree structure in addition to their score. For instance, an element that has many of its descendants retrieved (overlapping elements), but which are evenly distributed in the corresponding tree structure, and in addition have a similar score to the parent element, would be selected as the best element; otherwise its descendants would be considered instead. This approach was shown to be better than when selecting the highest scored element from each path as best elements.

A third type of approaches argue that the highest ranked elements may not necessarily be the most useful units of retrieval. For instance, a small but highly scored element may not be that useful. [35] ranks elements using a utility function that is based not only on the relevance score of an element, but its size, and the amount of irrelevant information contained in its children elements. An element whose utility value is higher to that of the sum of the utility value of its children is selected as best element. otherwise, the children elements whose utility values are above some threshold are selected.

Structure Constraints Since the beginning of INEX, there has been a debate on how to interpret structural constraints [49]. Initially, INEX required those constraints to be strictly matched. However, specifying an information need is not an easy task, in particular for semi-structured data with a wide variety of tag names. Although users may think they have a clear idea of the structural properties of the collection, there are likely to be aspects to which they are unaware. Obviously, this is very dependent on the actual application. Because of this, a vague interpretation of the structural constraints is followed: an element is relevant if it satisfies the information need, irrespective of the structural constraints, which are mainly considered as structural hints. That is, the aim of XML search system is to return components that contain the information sought after by the user even if the result elements do not exactly meet the structural conditions expressed in the query.

A common approach in INEX has been to manually build a dictionary of tag synonyms (e.g. "p" and "ip1" are considered equivalent) [34, 40]. [35] uses past relevance assessments for CAS queries to automatically build the dictionary. When a CAS topic asks for "section" element, all types of elements assessed relevant for that topic are added to the "section" synonym list. Other approaches decide to ignore the structure constraints in support elements, target elements, or both [9].

Scoring according to structural constraints has also been considered. In [22, 45], elements are first scored on how well the content constraints are satisfied, and the scores are boosted on the basis of matching the structural constraints. There have also been approaches in DB for relaxing structural constraints in XPath (e.g., [6, 41]). The framework in [6] expands query answers by incorporating approximate answers obtained using simple query relaxations. Such relaxations include tag relaxation as in INEX, making a child constraint a descendant constraint,

making an element in the query optional and, promoting content conditions to ancestor elements in the query. The framework also defines properties of scoring methods which guarantee that the score of an approximate answer is no higher than the score of an exact answer to a query.

5 Conclusion

We presented an overview of the challenges of designing query languages and scoring methods for XML search and a description of the INEX [20] effort, an initiative for the evaluation of XML retrieval methods. A major challenge for XML search is the combination of the effective retrieval of XML document components from the IR community with efficient query evaluation from the DB community. This challenge calls for a tighter integration of indices on both structure and content and acformalization of queries, in terms of a score-aware algebra, for the joint optimization of queries on both structure and text. An initial effort in this direction can be found in [12] and algorithms for the efficient evaluation of queries on both text and structure can be found in [4, 5, 46]. A second major challenge is the provision of a uniform and controllable platform to study all proposed scoring approaches and their various parameters, as they are so diverse (due to the richness of the XML mark-up and expected user behaviors) so that to obtain fundamental results regarding the best practices in XML scoring.

References

[1] S. Al-Khalifa, C. Yu, H. V. Jagadish. Querying Structured Text in an XML Database. SIGMOD 2003.

[2] S. Amer-Yahia, C. Botev, J. Shanmugasundaram. TeXQuery: A Full-Text Search Extension to XQuery. WWW 2004.

[3] S. Amer-Yahia, P. Case, T. Rölleke, J. Shanmugasundaram, G. Weikum Report on the DB/IR Panel at SIGMOD 2005. http://www.sigmod.org/sigmod/record/issues/0512/p71-column-cooper-1.pdf

[4] S. Amer-Yahia, E. Curtmola, A. Deutsch. Efficient XML Search with Complex Full-Text Predicates. SIGMOD 2006 (To appear).

[5] S. Amer-Yahia, N. Koudas, A. Marian, D. Srivastava, D. Toman. Content and Structure Scoring for XML. VLDB 2005.

[6] S. Amer-Yahia, L. Lakshmanan, S. Pandit. FleXPath: Flexible Structure and Full-Text Querying for XML. SIGMOD 2004.

[7] S. Amer-Yahia, M. Lalmas. Accessing XML Content: From DB and IR Perspectives (Tutorial). CIKM 2005.

[8] S. Amer-Yahia, J. Shanmugasundaram. XML Full-Text Search: Challenges and Opportunities (Tutorial). VLDB 2005.

[9] P. Arvola, J. Kekalainen, M. Junkkari. Query Evaluation with Structural Indices. INEX 2005.

[10] A. Balmin, V. Hristidis, N. Koudas, Y. Papakonstantinou, D. Srivastava, T. Wang. A System for Keyword Proximity Search on XML Databases. VLDB 2003.

[11] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, S. Sudarshan. Keyword Searching and Browsing in Databases using BANKS. ICDE 2002.

[12] C. Botev, S. Amer-Yahia, J. Shanmugasundaram. Expressiveness and Performance of Full-Text Search. EDBT 2006.

[13] J. M. Bremer, M. Gertz. XQuery/IR: Integrating XML Document and Data Retrieval. WebDB 2002.

[14] D. Carmel, Y. S. Maarek, M. Mandelbrod, Y. Mass, A. Soffer. Searching XML Documents via XML Fragments. SIGIR 2003.

[15] T. T. Chinenyanga, N. Kushmerick. Expressive and Efficient Ranked Querying of XML Data. WebDB 2001.

[16] C. Clarke. Controlling Overlap in Content-Oriented XML Retrieval. SIGIR 2005.

[17] S. Cohen, J. Mamou. Y. Kanza, Y. Sagiv. XSEarch: A Semantic Search Engine for XML. VLDB 2003.

[18] M. P. Consens, T. Milo. Algebras for Querying Text Regions: Expressive Power and Optimization. J. Comput. Syst. Sci. 57(3): 272-288 (1998)

[19] N. Fuhr, K. Grossjohann. XIRQL: An Extension of XQL for Information Retrieval. SIGIR 2000.

[20] N. Fuhr, M. Lalmas, Saadia Malik, Z. Szlávik. Advances in XML Information Retrieval. Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004

[21] N. Fuhr, T. Rölleke. A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems. ACM TOIS 15(1), 1997.

[22] S. Geva. GPX - Gardens Point XML IR at INEX 2005. INEX 2005.

[23] T. Grabs, H. Schek ETH Zürich at INEX: Flexible Information Retrieval from XML with PowerDB-XML. INEX Workshop 2002.

[24] L. Guo, F. Shao, C. Botev, J. Shanmugasundaram. XRANK: Ranked Keyword Search over XML Documents. SIGMOD 2003.

[25] V. Hristidis, L. Gravano, Y. Papakonstantinou. Efficient IR-Style Keyword Search over Relational Databases. VLDB 2003.

[26] J. Kamps, M. de Rijke, and B. Sigurbjornsson. Length normalization in xml retrieval. SIGIR 2004

[27] R. Kaushik, R. Krishnamurthy, J. F. Naughton, R. Ramakrishnan. On the Integration of Structure Indexes and Inverted Lists. ICDE 2004.

[28] G. Kazai, M. Lalmas INEX 2005 Evaluation Metrics. INEX 2005

[29] G. Kazai, M. Lalmas, A. P. de Vries. The Overlap Problem in Content-Oriented XML Retrieval Evaluation. SIGIR 2004

[30] Y. Li, C. Yu, H. V. Jagadish. Schema-Free XQuery. VLDB 2004.

[31] Library of Congress. http://lcweb.loc.gov/crsinfo/xml/.

[32] S. Liu, Q. Zou, and W. W. Chu. Configurable indexing and ranking for xml information retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 88–95, 2004.

[33] Y. Mass, M. Mandelbrod. Retrieving the most relevant XML Components. INEX 2004.

[34] Y. Mass, M. Mandelbrod. Using the INEX environment as a test bed for various user models for XML Retrieval. INEX 2005.

[35] V. Mihajlovic, G. Ramirez, T. Westerveld, D. Hiemstra, H. E. Blok, A. P. de Vries. TIJAH Scratches INEX 2005: Vague Element Selection, Image Search, Overlap, and Relevance Feedback. INEX 2005.

[36] P. Ogilvie, J. Callan. Hierarchical Language Models for XML Component Retrieval. INEX 2004.

[37] P. Ogilvie, J. Callan. Parameter Estimation for a Simple Hierarchical Generative Model for XML Retrieval. INEX 2005.

[38] B. Piwowarski, M. Lalmas. Providing Consistent and Exhaustive Relevance Assessments for XML Retrieval Evaluation. CIKM 2004.

[39] A. Salminen, F. Tompa. PAT Expressions: an Algebra for Text Search. Acta Linguistica Hungar. 41 (1-4), 1992

[40] K. Sauvagnat, L. Hlaoua, M. Boughanem XFIRM at INEX 2005: ad-hoc and relevance feedback tracks INEX 2005.

[41] T. Schlieder. Similarity Search in XML Data using Cost-Based Query Transformations. ACM SIGMOD 2001 Web and Databases Workshop.

[42] A. Schmidt, M. Kersten, M. Windhouwer. Querying XML Documents Made Easy: Nearest Concept Queries. ICDE 2001.

[43] B. Sigurbjörnsson, J. Kamps, M. de Rijke The Effect of Structured Queries and Selective Indexing on XML Retrieval. INEX 2005.

[44] B. Sigurbjörnsson, J. Kamps, M. de Rijke The Importance of Length Normalization for XML Retrieval. Journal of Information Retrieval, Vol. 8, Nbr 4, 2005.

[45] M. Theobald, R. Schenkel, G. Weikum. TopX & XXL at INEX 2005. INEX 2005.

[46] M. Theobald, R. Schenkel, G. Weikum. An Efficient and Versatile Query Engine for TopX Search. VLDB 2005.

[47] A. Theobald, G. Weikum. The Index-Based XXL Search Engine for Querying XML Data with Relevance Ranking. EDBT 2002.

[48] A. Trotman and B. Sigurbjörnsson. NEXI, Now and Next. INEX 2004.

[49] A. Trotman, M. Lalmas. The Interpretation of CAS. INEX 2005.

[50] J.-N. Vittaut, P. Gallinari. Machine Learning Ranking and INEX 05. INEX 2005.

[51] J.N. Vittaut, B. Piwowarski, P. Gallinari. An Algebra for Structured Queries in Bayesian Networks. INEX 2004.

[52] The World Wide Web Consortium. XQuery 1.0: An XML Query Language. W3C Working Draft. http://www.w3.org/TR/xquery/.

[53] The World Wide Web Consortium. XML Path Language (XPath) 2.0. W3C Working Draft. http://www.w3.org/TR/xpath20/.

[54] The World Wide Web Consortium. XQuery 1.0 and XPath 2.0 Full-Text. Working draft. http://www.w3.org/TR/xquery-full-text/.

[55] Y. Xu, Y. Papakonstantinou. Efficient Keyword Search for Smallest LCAs in XML Databases. SIGMOD 2005.

[56] C. Zhang, J. Naughton, D. DeWitt, Q. Luo, G. Lohman. On Supporting Containment Queries in Relational Database Management Systems. SIGMOD 2001.