XML Retrieval

Mounia Lalmas, Department of Computer Science, Queen Mary, University of London, United Kingdom, mounia@acm.org

Andrew Trotman, Department of Computer Science, University of Otago, New Zealand, andrew@cs.otago.ac.nz

SYNONYM

structured document retrieval, structured text retrieval, semi-structured text retrieval, focused retrieval, content-oriented XML retrieval

DEFINITION

Text documents often contain a mixture of structured and unstructured content. One way to format this mixed content is according to the adopted W3C standard for information repositories and exchanges, the *eXtensible Mark-up Language* (XML)[1]. In contrast to HTML, which is mainly layout-oriented, XML follows the fundamental concept of separating the *logical structure*[2] of a document from its layout. This logical document structure can be exploited to allow a more focused sub-document retrieval.

XML retrieval breaks away from the traditional retrieval unit of a document as a single large (text) block and aims to implement *focused retrieval* strategies aiming at returning document components, i.e. XML elements, instead of whole documents in response to a user query. This focused retrieval strategy 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 user's effort to locate relevant content within a document can be reduced by directing them to the most relevant parts of the document.

HISTORICAL BACKGROUND

Managing the enormous amount of information available on the web, in digital libraries, in intranets, and so on, requires efficient and effective indexing and retrieval methods. Although this information is available in different forms (text, image, speech, audio, video etc), it remains widely prevalent in text form. Textual information can be broadly classified into three categories depending on its level of structuring: structured, unstructured and semi-structured.

Structured information follows a pre-defined format, similar to a database table. The format of each record of a table is fixed and unambiguously defined by a database schema (e.g. the type, length and other attributes of each record). In contrast, unstructured information has no fixed pre-defined format, and is typically expressed in natural language. For instance, much of the information available on the web is unstructured. Although this information is mostly formatted in HTML, thus imposing some structure on the text, the structure is only for presentation purposes and carries essentially no semantic meaning. Correct nesting of the HTML structure (that is, to form an unambiguous document logical structure) is not imposed. Accessing structured information requires powerful but non-flexible query languages, such as SQL, whereas accessing unstructured information is through flexible but mostly simplistic means, such as a simple keyword matching or bag of words techniques.

Semi-structured information lies between structured and unstructured information. It has a stringent structure, but this structure may not be as rigid as a database schema. Semi-

[1] See entry on XML.
[2] See entry of Logical structure.

structured information is usually represented using XML, a mark-up language similar to HTML except that it imposes a rigorous structure on the document. Moreover, unlike HTML, XML tags are used to specify semantic information about the stored content and not the presentation. A document correctly marked-up in XML has a fixed document structure in which semantically separate document parts are explicitly identified – and this can be exploited to provide powerful and flexible access to textual information.

XML has been accepted by the computing community as a standard for document mark-up and an increasing number of documents are being made available in this format. As a consequence numerous techniques are being applied to access XML documents including information retrieval. The use of XML has generated a wealth of issues that are being addressed by both the database and information retrieval communities [BGS+03]. This entry is concerned with content-oriented XML retrieval [FL05, BFM06] as investigated by the information retrieval community. It does not deal with data-centric XML retrieval, which is being investigated by the database community.

Retrieval approaches for semi-structured text (marked-up in XML-like languages such as SGML) were first proposed in the late 1980s[3]. In the late 1990s, the interest in semi-structured text retrieval grew due to the introduction of XML in 1998. Research on XML information retrieval was first coordinated in 2002 with the founding of the Initiative for the Evaluation of XML Retrieval[4] (INEX). INEX provides a forum for the evaluation of information retrieval approaches specifically developed for XML retrieval.

SCIENTIFIC FUNDAMENTALS

Within INEX, the aim of an XML retrieval system is "to exploit the logical structure of XML documents to determine the best document components, i.e. best XML elements, to return as answers to queries" [KGL+03]. Query languages have been developed in order to allow users to specify the nature of these best components. Indexing strategies have been developed to obtain a representation not only of the content of XML documents, but their structure. Ranking strategies have been developed to determine the best elements for a given query.

Query languages

In XML retrieval the logical document structure is additionally used to determine which document components are most meaningful to return as query answers. With appropriate query languages, this structure can be specified by the user. For example, "I want a paragraph discussing penguins near to a picture labelled Otago Peninsula". Here, "penguins" and "Otago Peninsula" specify *content* (textual) constraints, whereas "paragraph" and "picture" specify *structural* constraints on the retrieval units.

Query languages for XML retrieval can be classified into content-only and content-and-structure query languages. Content-only queries[5] have historically been used as the standard form of input in information retrieval. They are suitable for XML search scenarios where user does not know (or is not concerned with) the logical structure of a document. Although only the content aspect of an information need can be specified, XML retrieval systems must still determine the best granularity of elements to return to the user.

Content-and-structure queries[6] provide a means for users to specify conditions referring both to the *content* and the *structure* of the sought elements. These conditions may refer to the content of specific elements (e.g. the returned element must contain a section about a particular topic), or may specify the type of the requested answer elements (e.g. sections

[3] See entry on Structured text retrieval models.
[4] See entry on INitiative for the Evaluation of XML retrieval (INEX).
[5] See entry on Content-only query.
[6] See entry on Content-and-structure query.

should be retrieved). There are three main categories of content-and-structured query languages [AL06]:

- Tag-based queries allow users to annotate words in the query with a single tag name that specifies the type of results to be returned. For example section:penguins requests section elements on "penguins''.
- Path-based queries are based upon the syntax of XPath[7]. They encapsulate the document structure in the query. An example in the NEXI[8] language is: //document[about(., Otago Peninsula)]//section[about(.//title, penguins)]. This query asks for sections that have a title about "penguins", and that are contained in a document about "Otago Peninsula".
- Clause-based queries use nested clauses to express information needs, in a similar way to SQL. The most prominent clause-based language for XML retrieval is XQuery[9]. A second example is XQuery Full-Text, which extends XQuery with text search predicates such as proximity searching and relevance ranking[10].

The complexity and the expressiveness of content-and-structure query languages increases from tag-based to clause-based queries. This increase in expressiveness and complexity often means that content-and-structured queries are viewed as too difficult for end users (because, they must, for example, be intimate with the document structure). Nonetheless they can be very useful for expert users in specialized scenarios, and also have been used as an intermediate between a graphical query language (such as Bricks [ZBO+06]) and an XML search engine.

Indexing strategies

Classical indexing methods in information retrieval make use of term statistics to capture the importance of a term in a document; and consequently for discriminating between relevant and non-relevant content. Indexing methods for XML retrieval require similar terms statistics, but for each element. In XML retrieval there are no a priori fixed retrieval units. The whole document, one of its sections, or a single paragraph within a section, all constitute potential answers to a single query. The simplest approach to allow the retrieval of elements at any level of granularity is to index each element separately (as a separate document in the traditional sense). In this case term statistics[11] for each element are calculated from the text of the element and all its descendants.

An alternative is to derive the term statistics through the aggregation of term statistics of the element own text, and those of each of its children[12]. A second alternative is to only index leaf elements and to score non-leaf elements through propagation of the score of their children elements. Both alternatives can include additional parameters incorporating, for instance, element relationships[13] or special behavior for some element types[14].

It is not uncommon to discard elements smaller than some given threshold. A single italicized word, for example, may not be a meaningful retrieval unit. A related strategy, selective indexing, involves building separate indexes for those element types previously seen to carry relevant information (sections, subsections, etc, but not italics, bold, etc.). With selective indexing the results from each index must be merged to provide a single ranked result list across all element types.

---

[7] See entry on XPath/XQuery.
[8] See entry on Narrowed Extended XPath I (NEXI).
[9] See entry on XPath/XQuery.
[10] See entry on XQuery Full-Text.
[11] See entry for Term statistics for semi-structured text retrieval.
[12] See entry on Aggregation-based semi-structured text retrieval
[13] See entry on Relationships in semi-structured text retrieval.
[14] See entry on Structure weight.

It is not yet clear which indexing strategy is the best. The best approach appears to depend on the collection, the types of elements (i.e., the DTD) and their relationships. In addition, the choice of the indexing strategy currently has an effect on the ranking strategy. More details about indexing strategies can be found in the entry on Indexing Units.

Ranking strategies

XML documents are made of XML elements, which define the logical document structure. Thus sophisticated ranking strategies can be developed to exploit the various additional (structural) evidence not seen in unstructured (flat) text documents.

Element scoring

Many of the retrieval models[15] developed for flat document retrieval have been adapted for XML retrieval. These models have been used to estimate the relevance of an element based on the evidence associated with the element only. This is done by a scoring function based, for instance, on the vector space, BM25, the language model, and so on. They are typically adapted to incorporate XML-specific features. As an illustration, we describe such a scoring function based on language models [KRS05].

Given a query $q = t_1, \cdots, t_n$ made of *n* terms t_i, an element *e* and its corresponding element language model θ_e, the element *e* is ranked using the following scoring function:

$$P(e \mid q) \propto P(e) \times P(q \mid \theta_e)$$

where $P(e)$ is the prior probability of relevance for element *e* and $P(q \mid \theta_e)$ is the probability of the query *q* being "generated" by the element language model and is calculated as follows:

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

Here $P(t_i \mid e)$ is the maximum likelihood estimate of term t_i in element *e*, $P(t_i \mid C)$ is the probability of query term t_i in the collection, and λ is the smoothing parameter. $P(t_i \mid e)$ is the element model based on element term frequency, whereas $P(t_i \mid C)$ is the collection model based on inverse element frequency. An important XML-specific feature is element length, since this can vary radically – for example, from a title to a paragraph to a document section. Element length can be captured by setting $P(e)$, the prior probability, as follows:

$$P(e) = \frac{length(e)}{\sum_{C} length(e)}$$

length(*e*) is the length of element *e*. Including length in the ranking calculation has been shown to lead to more effective retrieval than not doing so.

Contextualization

The above strategy only scores an element based on the content of the element itself. Considering additional evidence has shown to be beneficial for XML retrieval. In particular for long documents, using evidence from the element itself as well as its context (for example the parent element) has shown to increase retrieval performance. This strategy is referred to as contextualization[16]. Combining the element score and a separate document score has also been shown to improve performance.

[15] See entry for Information retrieval models.
[16] See entry on Contextualization.

Propagation
When only leaf elements are indexed, a propagation mechanism[17] is used to calculate the relevance score of the non-leaf elements. The propagation combines the retrieval scores of the leaf elements (often using a weighted sum) and any additional element characteristics (such as the distance between the element and the leaves). A non-trivial issue is the estimation of the weights for the weighted sum.

Merging
It has also been common to obtain several ranked lists of results, and to merge them to form a single list. For example, with the selective indexing strategy [MM05], a separate index is created for each a priori selected type of element (such as article, abstract, section, paragraph, and so on). A given query is then submitted to each index, each index produces a separate list of elements, normalization is performed (to take into account the variation in size of the elements) and the results are merged. Another approach to merging produces several ranked lists from a single index and for all elements in the collection (a single index is used as opposed to separate indices for each element). Different ranking models are used to produce each ranked list. This can be compared to document fusion investigated in the 1990s.

Processing structural constraints
Early work in XML retrieval required structural constraints in content-and-structure queries to be strictly matched but specifying an information need including structural constraints is difficult; XML document collections have a wide variety of tag names. INEX now views structural constraints as hints as to where to look (what sort of elements might be relevant). Simple techniques for processing structural constraints include the construction of a dictionary of tag synonyms and structure boosting. In the latter, the retrieval score of an element is generated ignoring the structural constraint but is then boosted if the element matches the structural constraint. More details can be found in the entry on Processing Structural Constraints.

Processing overlaps
It is one task to provide a score expressing how relevant an element is to a query but a different task to decide which of a set of several overlapping relevant elements is the best answer. If an element has been estimated relevant to a query, it is likely that its parent element is also estimated relevant to the query as these two elements share common text. But, returning chains of elements to the user should be avoided to ensure that the user does not receive the same text several times (one for each element in the chain). Deciding which element to return depends on the application and the user model. For instance, in INEX, the best element is one that is highly relevant, but also specific to the topic of request (i.e., does not discuss unrelated topics).

Removing overlap has mostly been done as a post-ranking task. A first approach, and the most commonly adopted one, is to remove elements directly from the result list. This is done by selecting the highest ranked element in a chain and removing any ancestors and descendents in lower ranks. Other techniques looked at the distribution of retrieved elements within each document to decide which ones to return. For example, the root element would be returned if all retrieved elements were uniformly distributed in the document. This technique was shown to outperform the simpler techniques. Efficiency remains an issue as the removal of overlaps is done at query time. More details can be found in the entry on Processing Overlaps.

KEY APPLICATIONS

[17] See entry Propagation-based semi-structured text retrieval

XML retrieval approaches (from query languages to ranking strategies) are relevant to any applications concerned with the access to repositories of documents annotated in XML, or similar mark-up languages such as SGML or ASN.1. Existing repositories include electronic dictionaries and encyclopedia such as the Wikipedia [DG06], electronic journals such as the journals of the IEEE [KGL+03], plays such as the collected works of William Shakespeare [KLR03], and bibliographic databases such as PubMed[18]. XML retrieval is becoming increasingly important in all areas of information retrieval, the application to full-text book searching is obvious and such commercial systems already exist [PT07].

EXPERIMENTAL RESULTS

Since 2002 work on XML retrieval has been evaluated in the context of INEX. Many of the proposed approaches have been presented at the yearly INEX workshops, held in Dagsthul, Germany. Each year, the INEX workshop pre-proceedings (which are not peer-reviewed) contain preliminary papers describing the details of participant's approaches. Since 2003 the final INEX workshop proceedings have been peer-reviewed, and since 2004 they have been published by Springer as part of the Lecture Notes in Computer Science series. Links to the pre- and final proceedings can be found on the INEX web site (http://www.inex.otago.ac.nz/).

DATA SETS

Since 2002 INEX has collected data sets that can be used for conducting XML retrieval experiments [LT07]. Each data set consists of a document collection, a set of topics, and the corresponding relevance assessments. The topics and associated relevance assessments are available on the INEX web site (http://www.inex.otago.ac.nz/)[19]. It should be noted that the relevance assessments on the latest INEX data set are released first to INEX participants.

CROSS REFERENCE

Aggregation-based semi-structured text retrieval
Content-and-structure query
Content-and-structure query
Evaluation metrics for semi-structured text retrieval
Indexing units
Information retrieval models
INitiative for the Evaluation of XML retrieval (INEX)
Integrated DB&IR semi-structured text retrieval
Logical structure
Narrowed Extended XPath I (NEXI)
Presenting semi-structured text retrieval results
Processing overlaps
Processing structural constraints
Propagation-based semi-structured text retrieval
Relationships in semi-structured text retrieval
Semi-structured text
Structure weight
Structured document retrieval
Structured text retrieval models
Term statistics for semi-structured text retrieval
XML
XQuery Full-Text
XPath/XQuery

[18] www.ncbi.nlm.nih.gov/pubmed/

[19] See the entry INitiative for the Evaluation of XML retrieval (INEX) for more details about evaluation methodology.

RECOMMENDED READING

[AL06] S. Amer-Yahia and M. Lalmas. XML Search: Languages, INEX and Scoring, *SIGMOD Record*, 35(4):16-23, 2006.

[BFM06] R. Baeza-Yates, N. Fuhr and Y.S. Maarek (Eds). Special Issue on XML retrieval, *ACM Transactions on Information Systems*, 24(4), 2006.

[BGS+03] H.M. Blanken, T. Grabs, H.-J. Schek, R. Schenkel and G. Weikum (Eds.). *Intelligent search on XML data, applications, languages, models, implementations, and benchmarks*, Springer, 2003.

[DG06] L. Denoyer and P. Gallinari. The Wikipedia XML corpus, *Comparative Evaluation of XML Information Retrieval Systems, 5th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2006, Revised and Selected Papers*, pp 12-19, 2007.

[FL05] N. Fuhr and M. Lalmas (Eds). Special Issue on INEX, *Information Retrieval* 8(4), 2005.

[KRS05] J. Kamps, M. de Rijke and B. Sigurbjörnsson. The Importance of Length Normalization for XML Retrieval, *Information Retrieval* 8(4): 631-654, 2005.

[KLR03] G. Kazai, M. Lalmas and J. Reid. Construction of a test collection for the focussed retrieval of structured documents, *25th European Colloquium on Information Retrieval Research, ECIR 2003*, Pisa, Italy, pp 88-103, April 2003.

[KGL+03] G. Kazai, N. Gövert, M. Lalmas and N. Fuhr. The INEX Evaluation Initiative, *Intelligent search on XML data, applications, languages, models, implementations, and benchmarks*, pp 279-293, 2003.

[LT07] M. Lalmas and A. Tombros. INEX 2002 - 2006: Understanding XML Retrieval Evaluation, *First International DELOS Conference, Pisa, Italy, Revised Selected Papers*, pp 187-196, 2007.

[MM05] Y. Mass and M. Mandelbrod. Component ranking and automatic query refinement for XML retrieval. *International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Revised Selected Papers*, pp 73-84, 2005.

[PT07] N. Pharo and A. Trotman. The use case track at INEX 2006. *SIGIR Forum* 41(1): 64-66, 2007.

[ZBO+06] R. van Zwol, J. Baas, H. van Oostendorp and F. Wiering. Bricks: The Building Blocks to Tackle Query Formulation in Structured Document Retrieval. *Advances in Information Retrieval, 28th European Conference on IR Research, ECIR 2006*, pp 314-325, 2006.