A Taxonomy for XML Retrieval Use Cases

Miro Lehtonen[1], Nils Pharo[2], and Andrew Trotman[3]

[1] Department of Computer Science, University of Helsinki, Finland
Miro.Lehtonen@cs.Helsinki.Fi

[2] Faculty of Journalism, Library and Information Science, Oslo University College, Norway
nils.pharo@jbi.hio.no

[3] Department of Computer Science, University of Otago, Dunedin, New Zealand
andrew@cs.otago.ac.nz

Abstract. Despite the active research on XML retrieval, it is a great challenge to determine the contexts where the methods can be applied and where the proven results hold. Therefore, having a common taxonomy for the use cases of XML retrieval is useful when presenting the scope of the research. The taxonomy also helps us design more focused user studies that have an increased validity. In the current state, the taxonomy covers most common uses of applying Information Retrieval to XML documents. We are delighted to see how some of the use cases match the tasks of the INEX participants.

1 Introduction

Research on XML IR methodology often lacks a clearly defined scope. Some researchers claim that their methods are general and applicable to arbitrary XML documents while others are accused of fine-tuning their methods to a single document collection. Despite the often supportive evidence presented with the results, these claims have no common theoretical grounding. Thus, we cannot describe how general each method is or whether a method is fine-tuned to a single collection, document type, document genre, or something else. The original contribution of this paper is the first proposal for a common taxonomy for the use cases of XML Retrieval. The taxonomy should help us see which aspects of XML retrieval are addressed and which use cases are potentionally involved in each research effort. Knowing those, it is quite straightforward to describe how methods generalise and how they are fine-tuned.

There are several approaches to creating a hierarchical classification for the use cases, depending on which features or facets we consider important. Therefore, we need to analyse the properties and characteristics of various use cases in order to find the right level of description. The factors that distinguish use cases from each other include the following:

XML A continuum from marked-up text including articles, books, web pages to structured data extracted from a database. The element names shift from presentation-specific names towards metadata-like content descriptors as we move marked-up text towards the other end of the continuum.

Content type Books and other monographies, chapters, articles, web pages, messages, document fragments and other incomplete documents.

Size A continuum from very small to very big. There is often interdepency between the content type and document size. The size usually makes a difference when presented to the user. Some documents are too big, whereas others may be too small, and the system has to work accordingly.

Queries Content-Only (CO), Content-And-Structure (CAS). Depending on the kind of XML, the document structure can be present in the queries as structural hints or requirements. Allowing XML structure in the queries usually requires user interfaces that support such querying.

Information need Answer to a question (QA), topic description, topic or value–based inventory of the document collection.

Presentation of search results Links to standalone documents optionally with relevant passages, best entry points, aggregated or assembled documents, heatmaps.

A similar effort was put forth by Andrei Broder who presented a taxonomy of web search [1]. He classified web queries into three classes according to their intent: navigational, informational, or transactional. While this classification of web searches is closely related to the user's task, the use cases of XML retrieval are tightly coupled with what kind of searching is possible, given certain XML documents. The contrast between the two classifications has a clear explanation: Anything can be searched on the web, whereas XML retrieval only concerns searching the contents of XML documents. For example, finding the URI of an XML document is not considered XML retrieval.

2 Hierarchical classification for use cases

One of the most important characteristics of XML is its ability to describe content with metadata which is a part of the markup. The tags around words, phrases, paragraphs, and even whole documents, define the interpretation of the enclosed content — only the interpreters vary. The purpose of the XML markup is the deciding factor at the top level of the hierarchy, so that in Class A, XML is computer-interpreted for display, in Class B, XML is human-readable, and in Class C, XML is interpreted and further processed by computer.

A Layout-oriented document types

XML documents of this nature are often conversions from other formats. In the past it has been common for academic publishers to use automated programs to generate the XML from typesetting files, whereas the legacy documents of enterprises are conversions from word processor formats or even HTML [2]. However, XML in both cases is the archival format: the XML "variant" of the document is definitive and all other formats are derivative.

XML: Most tag names describe either 1) the presentation of the content, e.g., `<i>` for italics, `<b>` for boldface, or 2) document structure, e.g., `<p>` for paragraph, `<sec>` for section. Differences between document types (DTDs, XML Schemas) are not a great challenge for IR applications where finding the relevant content comes first and publishing it comes later.

Queries: Keywords, keyphrases (Content-Only, CO). Including structural hints in the queries is not meaningful as far as element names are concerned. Layout and structure-related element names do not represent any information need as they are not related to the topic of any CO query.

Presentation of search results: The document order of the original documents is usually significant, so that the content of a single document or fragment cannot be reordered.

The size of the document is the second most distinctive feature in the class of layout-oriented documents as it directly affects the presentation of the search results as well as the overall system tasks. So far, we have identified three subcategories in Class A: Book search (A.1), Article search (A.2), and Fragment search (A.3).

A.1 Book search

One of the advantages of using XML-IR for book search (also for article search in A.2) is that the documents are long and XML-IR can locate relevant fragments for the user — reducing the load on the user. Although book search has not been explored by INEX researchers in the past, there might be a new track for book search at INEX 2007.

Example document types:

– Docbook[4].

Presentation of search results: Links (with summaries) to relevant sections and subsections or best entry points.

Example documents:

– Relevant chapters or sections of the book as standalone documents,
– Links (with summaries) to the full-text of whole books where the relevant content is highlighted.

User tasks: Learning and reading about a topic until information need is satisfied.

System tasks: Identifying relevant passages and best entry points.

[4] http://www.docbook.org/

A.2 Article search

The strength of XML shows in the ways the queries can be refined. For example, the set of matching articles can be limited to those that were published during a given time period, or those that were written by certain authors. Conditions can be set on any metadata included in the documents. Article search was on the agenda of the INEX initiatives 2002–2005.

Example documents:

- Web pages,
- IEEE journals in XML format (part of the INEX test suite 2002-2005).

Presentation of search results: Links (with summaries) to the full-text of whole articles. Optionally, the relevant content is highlighted.

User tasks: Learning and reading about a topic until information need is satisfied.

System tasks: Identifying relevant articles, relevant passages, and best entry points.

A.3 Fragment search

Having to search document fragments instead of whole documents is often the natural consequence of adopting an XML-based publishing environment. For example, the Franklin content management system of IBM decomposes information into reusable XML fragments [3]. Corresponding systems have become mainstream for enterprises. What is common to these environments is that the source documents are typically too small or too dependent on other documents (incomplete) to be published on their own. Whether the search application assembles the retrieved fragments into coherent answers is case-dependent.

Considering the number of operational systems, fragment search is one of the most common use cases of XML Retrieval. It is thus unfortunate that no INEX track has yet been devoted to it. Future prospects do not look any brighter as the realistic and often proprietary document collections are beyond reach.

Example documents:

- Single-sourced documentation [4].

Presentation of search results: Customised according to fragment size, content, and final publication format.

User tasks: Finding all relevant fragments to be included in a publication, e.g. technical document, or to be used as background material.

System tasks: Identifying relevant fragments and combining them into whole documents.

B Content-oriented document types

Adding metadata to documentation and making the documents self-contained were some of the major incentives for developing the XML standard. Those goals are fulfilled in content-oriented XML documents.

XML: The majority of the non-optional tag names describe the text content of the corresponding elements, e.g., `<action>`, `<reason>`, `<recommendation>`. Differences between document types (DTDs, XML Schemas) have a direct impact on the query evaluation which may require manual mappings between incompatible structures. Automatic and ontology-based methods are also worth considering.

Queries: Keywords, keyphrases, key values, and structure (Content-And-Structure, CAS). Without structural hints in the query, the results may be highly ambiguous, because the interpretation of the content is dependent on the XML names.

Presentation of search results: The relevant results are rarely presented in the original order even when they come from the same document. For each relevant answer, the content is ordered by rules or templates. Each XML document, in turn, may hold more content than what would ever be published as a single publication.

The second most distinctive feature of content-oriented document types is the level where the content is described. If the content-oriented element names are mostly at the level of paragraphs and sections, for example, `<instructions>` or `<weatherforecast>`, we may apply methods for *semantic search* [5] to the documents (B.1). If the content is mostly described at the inline-level, the typical element names include `<city>`, `<person>`, and `<code>`. The most appropriate style of search is then for entities (B.2). If all the element names describe the content, we are ready for data retrieval (B.3).

B.1 Semantic search

Semantic search is much like article search (A.2) to the user. The biggest difference between semantic search and article search is that the XML documents for the former contain semantic markup which the search application is aware of.

Example documents:

- Documentation with descriptive element names (metadata) and full-text content,
- The Guideline Elements Model (GEM) [6].

Presentation of search results:

– The relevant parts of the original documents reformatted according to the query.
– Standalone XML fragments including the relevant content which is extracted from the source document.

User tasks: Finding topic descriptions, learning and reading about the topic.

System tasks: Identifying relevant elements, vague matching of content and structural search conditions, organising the matches into whole documents.

B.2 Entity search

Full-text content where certain entities are marked up is a typical environment for entity search.

Example documents:

– Annotated documents

Presentation of search results: Standalone fragments, possibly formatted.

User tasks: Question asking.

System tasks: Question Answering: Identifying relevant elements, duplicate detection, vague matching of data (target elements, element and attribute names) and structural search conditions on them.

B.3 Data retrieval

Thanks to its interoperability, XML is the appropriate interchange format, for example between two different (relational) database formats. Consequently, most databases provide an XML view of the data that is equivalent to the native representation format and available for querying.

Example documents:

– XML databases, relational data for XML.

Presentation of search results: Template-driven formatting.

User tasks: Database publishing, template-driven document assembly.

System tasks: Identifying relevant elements.

C Process-oriented document types

Although the process-oriented XML documents are searched more often than any other types of XML documents, they are not paid much attention in the IR-related literature. Operational search applications have been available online for several years, though. The systems typically index and retrieve only XML documents, however, they have been sadly neglected by the XML Retrieval community. What is typical of searching process-oriented documents is that the query is matched against one part of the XML document (metadata), whereas other parts are expected as answers (data).

The applications that process the XML of Class C are highly aware of the XML markup of the documents. Good search applications are likely to follow the practice which is discouraging to the development of more general methods.

XML: Most of the tag names describe the technical interpretation of the content. Common tag names include `<sequence>`, `<input>`, `<operation>`, and `<port>`. Search applications typically specialise in a single set of tag names (a single document type), which shows in fine-tuned methods and tailored user interfaces.

Queries: Keywords, keyphrases, and structural constraints with domain-specific interpretation.

Presentation of search results: The relevant answers are listed as links pointing to the actual answers. Some answers such as Web services (C.3) are naturally viewed in a web browser, whereas others may need separate applications. For example, newsfeeds (C.2) are opened with a feed reader.

C.1 Multimedia search

As XML has also become a format for multimedia, it is natural to query and search such XML documents. However, the XML representation of multimedia is often the source format, whereas systems are more accustomed to querying the derivative formats such as jpg and pdf. Whether search applications will actually specialise in querying the XML of the multimedia documents is thus uncertain.

Example document types:

- SVG[5], SMIL[6].
- Other documents that may contain multimedia.

User tasks: Finding multimedia.

[5] http://www.w3.org/Graphics/SVG/
[6] http://www.w3.org/AudioVideo/

System tasks: Ranking multimedia by metadata descriptions, possibly interpret the XML structures, as well.

C.2 Feed search

Information filtering has long been and IR application [7], but it was not until XML came around that searching newsfeeds (including blogs) online had a real user demand. The first operational implementations of XML feed search date back to 2003 when Feedster launched their online system[7].

Example document types:

– RSS[8], OPML[9].

User tasks: Finding relevant postings.

System tasks: Indexing and querying streaming XML, monitoring and filtering XML feeds.

C.3 Web Services search

Finding the most relevant web services has become an application area of XML Retrieval, thanks to the XML-based Web Services Description Language (WSDL) [8]. Woogle[10] [9] is a decent implementation of a such a search.

Example document types:

– WSDL.

User tasks: Finding good web services that are compatible with their demand.

System tasks: Matching keywords with both service descriptions and operations, as well as input and output parameters. Compositions of operations with similar functionality to the search operations may also be returned.

C.4 Message search

When XML is sent in an XML envelope, it becomes an XML message. Systems that send and receive such messages also store them for archival purposes. Searching the archived XML messages is a potential application area of XML Retrieval.

[7] http://www.feedster.com/
[8] http://web.resource.org/rss/1.0/
[9] http://www.opml.org/spec
[10] http://www.cs.washington.edu/woogle

Example document types:

– SOAP [10].
– Other transactional documents.

User tasks: Finding relevant messages.

System tasks: Matching strings and values with the relevant parts of the message, indexing messages.

3 Conclusion

Creating the hierarchical classification for the use cases of XML retrieval is only a first step in developing the taxonomy. The next step is to describe each class in more detail, e.g., by identifying the challenges and opportunities of each class. Although the descriptions are still incomplete, we can learn a few key points from them:

– Both the user tasks and system tasks are different for each use case, which implies that no system can have a single search interface for all the use cases.
– Only few use cases are included in the INEX-related research, possible because of some diversity in the interpretation of "XML Retrieval".

The two-level classification described in this paper is the result of at least one round-table meeting, some email correspondence, literature review, a few interviews with two graduate students who independently develop systems for XML retrieval, and a few interviews with commercial representatives of the XML server industry. This work is on-going and the authors appreciate any feedback and contributions that advance this work.

References

1. Broder, A.: A taxonomy of web search. SIGIR Forum **36** (2002) 3–10
2. Chidlovskii, B., Fuselier, J.: Supervised learning for the legacy document conversion. In: DocEng '04: Proceedings of the 2004 ACM symposium on Document engineering, New York, NY, USA, ACM Press (2004) 220–228
3. Weitzman, L., Dean, S.E., Meliksetian, D., Gupta, K., Zhou, N., Wu, J.: Transforming the content management process at ibm.com. In: CHI '02: Case studies of the CHI2002/AIGA Experience Design FORUM, New York, NY, USA, ACM Press (2002) 1–15
4. Clark, D.: Rhetoric of present single-sourcing methodologies. In: SIGDOC '02: Proceedings of the 20th annual international conference on Computer documentation, New York, NY, USA, ACM Press (2002) 20–25
5. Chu-Carroll, J., Prager, J., Czuba, K., Ferrucci, D., Duboue, P.: Semantic search via XML fragments: a high-precision approach to IR. In: SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, ACM Press (2006) 445–452

6. ASTM International: E2210-02 Standard Specification for Guideline Elements Model (GEM)-Document Model for Clinical Practice Guidelines. (2003)
7. Belkin, N.J., Croft, W.B.: Information filtering and information retrieval: two sides of the same coin? Commun. ACM **35** (1992) 29–38
8. W3C: Web Services Description Language (WSDL) Version 2.0, W3C Candidate Recommendation. (27 March 2006) Latest version available at `http://www.w3.org/TR/wsdl20`.
9. Dong, X., Halevy, A.Y., Madhavan, J., Nemes, E., Zhang, J.: Similarity search for web services. In: VLDB '04: Proceedings of the 30th International Conference on Very Large Data Bases. (2004) 372–383
10. W3C: SOAP Version 1.2 Part 0: Primer (Second Edition), W3C Proposed Edited Recommendation. (19 December 2006) Lates version available at `http://www.w3.org/TR/soap12-part0/`.