XML Retrieval: state of the art and new challenges

Parzialmente tratto da “Advances in XML retrieval: The INEX Initiative” di Norbert Fuhr

Ing. Stefania Marrara
Introduction

I. XML Retrieval: Models & Methods

II. Views on XML Retrieval

III. Giving flexibility to XQuery
Part I: Models and methods for XML retrieval
Structured Document Retrieval

- Traditional IR is about finding relevant documents to a user’s information need, e.g. entire book.

- SDR allows users to retrieve document components that are more focussed to their information needs, e.g a chapter, a page, several paragraphs of a book instead of an entire book.

- The structure of documents is exploited to identify which document components to retrieve.

• Structure improves precision
XML retrieval allows users to retrieve document components that are more focussed, e.g. a subsection of a book instead of an entire book.
Queries

- Content-only (CO) queries
  - Standard IR queries but here we are retrieving document components
  - “London tube strikes”

- Structure-only queries
  - Usually not that useful from an IR perspective
  - “Paragraph containing a diagram next to a table”

- Content-and-structure (CAS) queries
  - Put constraints on which types of components are to be retrieved
    - E.g. “Sections of an article in the Times about congestion charges”
    - E.g. Articles that contain sections about congestion charges in London, and that contain a picture of Ken Livingstone, and return titles of these articles
  - Inner constraints (support elements), target elements
Return document components of varying granularity (e.g. a book, a chapter, a section, a paragraph, a table, a figure, etc), relevant to the user’s information need both with regards to content and structure.

SEARCHING = QUERYING + BROWSING
Conceptual model

Structured documents

Content + structure

Documents

Indexing

tf, idf, …

Document representation

Query

Formulation

Query representation

Retrieval function

Matching content + structure

Retrieval results

Presentation of related components
Challenge 1: term weights

No fixed retrieval unit + nested document components:
- how to obtain document and collection statistics (e.g. tf, idf)
- inner aggregation or outer aggregation?
**Challenge 2: augmentation weights**

Nested document components:
- which components contribute best to content of Article?
- how to estimate weights (e.g. size, number of children)?
Challenge 3: component weights

Different types of document components:
- which component is a good retrieval unit?
- is element size an issue?
- how to estimate component weights (frequency, user studies, size)?
Challenge 4: overlapping elements

Nested (overlapping) elements:
- Section 1 and article are both relevant to “XML retrieval”
- which one to return so that to reduce overlap?
- should the decision be based on user studies, size, types, etc?
Controlling Overlap

• Start with a component ranking, elements are re-ranked to control overlap.
• Retrieval status values (RSV) of those components containing or contained within higher ranking components are iteratively adjusted

1. Select the highest ranking component.
2. Adjust the RSV of the other components.
3. Repeat steps 1 and 2 until the top $m$ components have been selected.

(SIGIR 2005)
XML retrieval

- Efficiency: Not just documents, but all its elements
- Models
  - Statistics to be adapted or redefined
  - Aggregation / combination
- User tasks
  - Focussed retrieval
  - No overlap
  - Do users really want elements
- Link to web retrieval / novelty retrieval
- Interface and visualisation
- Clustering, categorisation, summarisation
- Applications
  - Intranet, the Internet(?), digital libraries, publishing companies, semantic web, e-commerce
Evaluation of XML retrieval: INEX

- Evaluating the effectiveness of content-oriented XML retrieval approaches

- Collaborative effort $\Rightarrow$ participants contribute to the development of the collection queries relevance assessments

- Similar methodology as for TREC, but adapted to XML retrieval
INEX test suites

- Corpora:
  - 16,819 articles in XML format from IEEE Computer Society (~750MB)
  - Wikipedia snapshot from April 2006 (660,000 articles, 4.6 GB)

- Queries:
  - 280 queries for IEEE-CS
  - 111 queries for Wikipedia

- Relevance judgments
  For the top 100 answers from each participant

- Collaborative effort:
  queries and relevance judgments from the 50-70 annual participants
Part II: Views on XML Retrieval
Views on XML

Element Typing

Object Types

Data Types

Text only

Structure

Nested structure

Named fields

XPath

XQuery
XML structure: 1. Nested Structure

- XML document as hierarchical structure
- Retrieval of elements (subtrees)
- Typical query language does not allow for specification of structural constraints
- Relevance-oriented selection of answer elements: return the most specific relevant elements
XML structure: 2. Named Fields

- Reference to elements through field names only
- Context of elements is ignored (e.g. author of article vs. author of referenced paper)
- Post-Coordination may lead to false hits (e.g. author name – author affiliation)
- [Kamps et al. (TOIS 4/06)]: XML retrieval quality does not suffer from restriction to named fields

Example: Dublin Core

```
  <dc:title>Generic Algebras</dc:title>
  <dc:creator>A. Smith (ESI), B. Miller (CMU)</dc:creator>
  <dc:subject>Orthogonal group, Symplectic group</dc:subject>
  <dc:date>2001-02-27</dc:date>
  <dc:format>application/postscript</dc:format>
  <dc:source>ESI preprints</dc:source>
  <dc:language>en</dc:language>
</oai_dc:dc>
```
XML structure: 3. XPath (cont’d)

- Full expressiveness for navigation through document tree (+links)
  - Parent/child, ancestor/descendant
  - Following/preceding, following-sibling, preceding-sibling
  - Attribute, namespace
- Selection of arbitrary elements
- Too complex for users?
XML structure: 4. XQuery

Higher expressiveness, especially for database-like applications:
- Joins
- Aggregations
- Constructors for restructuring results

Example: List each publisher and the average price of its books.

```xquery
FOR $p IN distinct(document("bib.xml")//publisher)
LET $a := avg(document("bib.xml")//book[publisher = $p]/price)
RETURN
<publisher>
  <name> {$p/text()} </name>
  <avgprice> {$a} </avgprice>
</publisher>
```

How many papers on digital libraries by Ed Fox?
This text explains all about XML and IR.

Example query:
```
//chapter[about(.,
   XML query language]
```
XML entity types: 2. Data Types

- Data type: domain + (vague) predicates
  - Language (multilingual documents) / (language-specific stemming)
  - Person names / “his name sounds like Jones”
  - Dates / “about a month ago”
  - Amounts / “orders exceeding 1 Mio $”
  - Technical measurements / “at room temperature”
  - Chemical formulas

- Close relationship to XML Schema, but
  - XMLS supports syntactic type checking only
  - No support for vague predicates
Pablo Picasso (October 25, 1881 - April 8, 1973) was a Spanish painter and sculptor..... In Paris, Picasso entertained a distinguished coterie of friends in the Montmartre and Montparnasse quarters, including André Breton, Guillaume Apollinaire, and writer Gertrude Stein.

To which other artists did Picasso have close relationships?

Did he ever visit the USA?

Named entity recognition methods allow for automatic markup of object types

Object types support increased precision
Tag semantics?

Object type hierarchies

- Person
  - Scientist
    - Physicist
    - Chemist
  - Artist
    - Poet
    - Actor
    - Singer

Role hierarchies

- Creator
  - Author
  - Editor
Conclusion and future work

- Research issues in XML retrieval
  - Effective retrieval of XML documents
  - What and how to evaluate

- Interactive XML retrieval:
  - Empirical foundation for the need for element retrieval (instead of full documents)

- Views on XML:
  - Large variety of possible applications
  - But lack of appropriate test collections

- XML and Semantic Web technologies:
  - Potentially useful, especially in limited domains (but open research issues)
Part III: Giving Flexibility to XQuery
XQuery is the new W3C standard for querying XML
Maybe difficult but extremely powerful
Many XML collections are created for experts (e.g., genoma descriptions, computer science journals and papers etc.)
Many experts are used to query their target collection and know – more or less - the general structure of the information stored

Why providing XML retrieval only for dummy users?
Why not recognizing that experts could appreciate XQuery expressing power?
**XQuery advantages and drawbacks**

- XQuery is a standard from W3C, many good query engines are today available (e.g. Galax)
- XQuery has a very good expressive power (FLOWR, functions etc.)
- XQuery can take into consideration both information in the node content and in the structure of the document

**BUT**

- XQuery can express only EXACT queries: structure and node content of the query expression must reflect those in the document to be retrieved or the result is empty
Questions

- What happens when the user has a partial or uncomplete knowledge of the structure and the content she is interested in?
- How can we help her in finding useful information using the partial knowledge she has without the restrictions of XQuery?
Giving flexibility to XQuery

- We propose a flexible XML query language whose search paths can contain both traditional and flexible constraints on structure and content, able to provide a query formulation in which the user can decide the degree and the type of flexibility in identifying the relevant information items.

- The XQuery extension we propose is based on a query evaluation process performed in two subsequent phases.
  - The first phase is aimed at performing content-based matching to reduce the number of XML structures to be later examined by (more costly) structure matching, greatly improving the overall performance of the index.
  - The second phase applies structure-based matching to the documents pre-selected by the first evaluation stage.

- The flexible constraints:
  - On content: cw, around,
  - On structure: below, near and approximately
The two-phase query evaluation process

- The two-phase query evaluation process is performed in four basic steps:
  - the keywords specified in the query leaf nodes are extracted;
  - secondly, if the query contains an approximate constraint on node content, keywords are enriched (i.e., if the query requires something similar to `butterfly' the set of keywords will include `butterfly' but also `butterflies');
  - the selected keywords are used to perform a traditional IR content-based retrieval. The documents resulting by this evaluation phase constitute the input to the subsequent structure evaluation phase;
  - finally the constraints on the document structure are evaluated on the pre-selected documents and the results are returned to the user.
The two-layers indexing structure

- Dictionary
  - Term $t$
  - NDOC$_t$

- Postings
  - docID
  - doc_occ
  - nodeID
  - occ

- XDG
  - node#
  - docIDs
  - positions
  - Physical
  - addresses

Path indexing file

Term inverted file
ERROR:
invalidrestore

OFFENDING COMMAND: restore

ERROR: invalidrestore