XML Retrieval

Norbert Fuhr
University of Duisburg-Essen

*with input from Mounia Lalmas, QMUL/Glasgow
Contents

I. Structured Document Retrieval
II. XML Retrieval Methods
III. Evaluation of XML Retrieval
IV. Views on XML Retrieval
Part I: Structured Document Retrieval
Structured Documents

- In general, any document can be considered structured according to one or more structure-type:
  - Linear order of words, sentences, paragraphs …
  - Hierarchy or logical structure of a book’s chapters, sections …
  - Links (hyperlink), cross-references, citations …
  - Temporal and spatial relationships in multimedia documents
Structured Documents

- The structure can be implicit or explicit
- Explicit structure is formalised through document representation standards (Mark-up Languages)
  - Layout
    - LaTeX (publishing), HTML (Web publishing)
  - Structure
    - SGML, XML (Web publishing, engineering), MPEG-7 (broadcasting)
  - Content/Semantic
    - RDF (ontology)
Structured document retrieval

- (1989-1995)

- Passage retrieval
- Structured text models
  - Hybrid model (flat fields)
  - PAT expressions
  - Overlapped lists
  - Reference lists
  - Proximal nodes
  - Region algebra
    - Proposed as Algebra for XML-IR-DB Sandwich
  - p-strings
  - Tree matching
Passage retrieval

- Passage: continuous part of a document, Document: set of passages

  doc

  p1          p2        p3                        p4                          p5           p6

- A passage can be defined in several ways:
  - Fixed-length e.g. (300-word windows, overlapping)
  - Discourse (e.g. sentence, paragraph) e.g. according to logical structure but fixed (e.g. passage = sentence, or passage = paragraph)
  - Semantic (TextTiling based on sub-topics)

- Apply IR techniques to passages
  - Retrieve passage or document based on highest ranking passage or sum of ranking scores for all passages
  - Deal principally with content-only queries

(Callan, SIGIR 1994; Wilkinson, SIGIR 1994; Salton et al, SIGIR 1993; Hearst & Plaunt, SIGIR 1993; …)
Region algebra

- Manipulates text intervals - “between which positions in the document?”; and uses containment relationships - “in which components?”
  - Various methods but with similar aims: Simple Concordance List, Generalised Concordance List, Proximal Nodes …

- Ranking based on word distances
- Suited for CO and CAS queries

Query: “document” and “retrieval”
Intervals: {(102, 103)(107, 108)}

Query: [chapter] containing SDR
Intervals: {((103.2, 167.2)}

(SIGIR 1992, but see also XML retrieval Mihajlovic et al CIKM 2005)
XML query languages (very brief)

- Four “levels” of expressiveness
  - Keyword search (CO Queries)
    - “xml”
  - Tag + Keyword search
    - book: xml
  - Path Expression + Keyword search (CAS Queries)
    - /book[./title about “xml db”]
  - XQuery + Complex full-text search
    - for $b in /book
      let score $s := $b ftcontains “xml” && “db”
      distance 5

(Amer-Yahia & Lalmas, SIGMOD Record 2006)
Part II: XML Retrieval Methods
XML retrieval vs. document retrieval

- No predefined unit of retrieval
- Dependency of retrieval units
- Aims of XML retrieval:
  - Not only to find relevant elements
  - But those at the appropriate level of granularity
Content-oriented XML retrieval = Focused Retrieval

XML retrieval allows users to retrieve document components that are more focused, e.g., a subsection of a book instead of an entire book.

Note:
Here, document component = XML element
Focused Retrieval: Principle

- A XML retrieval system should always retrieve the most specific part of a document answering a query.

- Example query: **football**

- Document
  
  ```xml
  <chapter> 0.3 football
    <section> 0.5 history </section>
    <section> 0.8 football 0.7 regulation </section>
  </chapter>
  ```

- Return `<section>`, not `<chapter>`
Challenge 1: Term statistics

No fixed retrieval unit + nested document components:
- how to obtain element and collection statistics (e.g. tf, idf)?
- which aggregation formalism to use?
- inner or outer aggregation?
Challenge 2: Relationship statistics

Relationship between elements:
- Which sub-element(s) contribute best to content of its parent element and vice versa?
- How to estimate (or learn) relationship statistics (e.g. size, number of children, depth, distance)?
- How to aggregate term and/or relationship statistics?
Challenge 3: Structure statistics

Different types of elements:
- which element is a good retrieval unit?
- is element size an issue?
- how to estimate (or learn) structure statistics (frequency, user studies, size, depth)?
- how to aggregate term, relationship and/or structure statistics?
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?
Challenge 5: Interpreting structural constraints

• Ideally:
  – There is one DTD/schema
  – User understands DTD/schema

• In practice: rare
  – Many DTs/schemas
  – DTDs/Schema not known in advance
  – DTDs/Schema change
  – Users do not understand DTDs/schema

• Need to identify “similar/synonym” elements/tags
• Strict or vague interpretation of the structure

• Relevance feedback/blind feedback?
Retrieval models …

divergence from randomness

vector space model

cognitive model

Boolean model

statistical model

logistic regression

structured text models

extending DB model

Bayesian network

machine learning

language model

belief model

probabilistic model

natural language processing

….
Retrieval units: What to Index?

- XML documents are trees with a hierarchical structure of nested elements (sub-trees).
- What should we put in the index?
  - There is no fixed unit of retrieval.
Retrieval units: XML sub-trees

Assume a document like

```xml
<article>
  <title>XXX</title>
  <abstract>YYY</abstract>
  <body>
    <sec>ZZZ</sec>
    <sec>ZZZ</sec>
  </body>
</article>
```

Index separately

- `<article>XXX YYY ZZZ ZZZ </article>`
- `<title>XXX </title>`
- `<abstract>YYY </abstract>`
- `<body>ZZZ ZZZ </body>`
- `<sec>ZZZ </sec>`
Retrieval units: XML sub-trees

- Indexing sub-trees is closest to traditional IR
  - each XML elements is bag of words of itself and its descendants
  - and can be scored as ordinary plain text document

- Advantage: well-understood problem

- Negative:
  - redundancy in index
  - terms statistics

- Led to the notion of indexing nodes
- Problem: how to select them?
  - manually, frequency, relevance data
(XIRQL) Indexing nodes

(Fuhr & Großjohann, SIGIR 2001)
Retrieval units: Disjoint elements

Assume a document like

```xml
<article>
  <title>XXX</title>
  <abstract>YYY</abstract>
  <body>
    <sec>ZZZ</sec>
    <sec>ZZZ</sec>
    <sec>ZZZ</sec>
  </body>
</article>
```

Index separately

- `<title>XXX</title>`
- `<abstract>YYY</abstract>`
- `<sec>ZZZ</sec>`
- `<sec>ZZZ</sec>`

Note that `<body>` and `<article>` have not been indexed
Retrieval units 2: Disjoint elements

- Main advantage and main problem
  - (most) article text is not indexed under /article
  - avoids redundancy in the index

- But how to score higher level (non-leaf) elements?
  - Propagation/Augmentation approach
  - Element specific language models
(Geva, INEX 2004, INEX 2005)

Propagation - GPX model

Leaf elements score

\[ L = N^{n-1} \sum_{i=1}^{n} \frac{t_i}{f_i} \]

Branch elements score

\[ RSV = D(n) \sum_{i=1}^{n} L_i \]

\( n \): the number of unique query terms

\( N \): a small integer (\( N=5 \), but any \( 10 > N > 2 \) works)

\( t_i \): the frequency of the term in the leaf element

\( f_i \): the frequency of the term in the collection

\( L_i \): child element score

\( D(n) = \) relationship statistics

- \( 0.49 \) if \( n = 1 \)
- \( 0.99 \) otherwise

\( D(n) \) scores are recursively propagated up the tree
Element specific language model (simplified)

Assume a document

\[ \langle \text{bdy} \rangle \]
\[ \langle \text{sec} \rangle \text{cat...}</\text{sec}\]
\[ \langle \text{sec} \rangle \text{dog...}</\text{sec}\]
\[ \langle /\text{bdy} \rangle \]

Query: cat dog

- **Assume**
  - \( P(\text{dog}|\text{bdy/sec[1]})=0.7 \)
  - \( P(\text{cat}|\text{bdy/sec[1]})=0.3 \)
  - \( P(\text{dog}|\text{bdy/sec[2]})=0.3 \)
  - \( P(\text{cat}|\text{bdy/sec[2]})=0.7 \)

- **Mixture**
  \[ P(w|e) = \sum \lambda_i P(w|e_i) \]
  - With uniform weights (\( \lambda=0.5 \))
  - \( \lambda = \text{relationship statistics} \)
  - \( P(\text{cat}|\text{bdy})=0.5 \)
  - \( P(\text{dog}|\text{bdy})=0.5 \)
  - So /bdy will be returned

(Ogilvie & Callan, INEX 2004)
Retrieval units: Distributed

- Index separately particular types of elements
- E.g., create separate indexes for
  - articles
  - abstracts
  - sections
  - subsections
  - subsubsections
  - paragraphs …

- Each index provides statistics tailored to particular types of elements
  - language statistics may deviate significantly
  - queries issued to all indexes
  - results of each index are combined (after score normalization)
Distributed: Vector space model

structure statistics

article index $\xrightarrow{\text{RSV}}$ normalised RSV

abstract index $\xrightarrow{\text{RSV}}$ normalised RSV

section index $\xrightarrow{\text{RSV}}$ normalised RSV

sub-section index $\xrightarrow{\text{RSV}}$ normalised RSV

paragraph index $\xrightarrow{\text{RSV}}$ normalised RSV

merge

tf and idf as for fixed and non-nested retrieval units

(Mass & Mandelbrod, INEX 2004)
Retrieval units: Distributed

- Only part of the structure is used
  - Element size
  - Relevance assessment
  - Others

- Main advantages compared to disjoint element strategy:
  - avoids score propagation which is expensive at run-time
  - index redundancy is basically pre-computing propagation
  - XML specific propagation requires nontrivial parameters to train

- Indexing methods and retrieval models are “standard” IR
  - although issue of merging - normalization
Combination: Language model

- Element language model
- Collection language model
- Smoothing parameter $\lambda$

Element score

- High value of $\lambda$ leads to increase in size of retrieved elements

Element size
- Element score
- Article score

Rank element

- Structure statistics
- Query expansion with blind feedback
- Ignore elements with $\leq 20$ terms

(Sigurbjörnsson et al., INEX 2003, INEX 2004)
Combination: Machine learning

- Use of standard machine learning to train a function that combines
  - Parameter for a given element type
  - Parameter * score(element)
  - Parameter * score(parent(element))
  - Parameter * score (document)

- Training done on relevance data (previous years)
- Scoring done using OKAPI

(Vittaut & Gallinari, ECIR 2006)
Combination: Contextualization

• Basic ranking by adding weight value of all query terms in element.

• Re-weighting is based on the idea of using the ancestors of an element as a context.
  • Root: combination of the weight of an element its \(1.5 \times \text{root}\).
  • Parent: average of the weights of the element and its parent.
  • Tower: average of the weights of an element and all its ancestors.
  • Root + Tower: as above but with \(2 \times \text{root}\).

• Here root is the document

(Arvola et al., CIKM 2005, INEX 2005)
Post-processing: Displaying XML Retrieval Results

• XML element retrieval is a core task
  • how to estimate the relevance of individual elements
• However, it may not be the end task
  • Simply returning a ranked list of elements results seems insufficient
    • may have overlapping elements
      • elements from the same article may be scattered
• This may be dealt with in special XML retrieval interfaces
  • Cluster results, provide heatmap, best entry point, …
New retrieval tasks (at INEX)

- INEX 2005-7 addressed two new retrieval tasks
  - Thorough is ‘pure’ XML element retrieval as before
  - Focused does not allow for overlapping elements to be returned
  - Fetch and Browse requires results to be clustered per article
    - Various variants

- New tasks require post-processing of ‘pure’ XML element runs
  - geared toward displaying them in a particular interface
Post-processing: Controlling Overlap

What most approaches are doing:

- Given a ranked list of elements:
  1. select element with the highest score within a path
  2. discard all ancestors and descendants
  3. go to step 1 until all elements have been dealt with

- (Also referred to as brute-force filtering)
“Post”-Processing: Removing overlap

- Sometimes with some “prior” processing to affect ranking:
  - Use of a utility function that captures the amount of useful information in an element
    
    Element score * Element size * Amount of relevant information

- Used as a prior probability

- Then apply “brute-force” overlap removal

(Mihajlovic et al., INEX 2005; Ramirez et al., FQAS 2006)
Post-Processing: Removing overlap

(Mass & Mandelbrod, INEX 2005)

Smart filtering
Given a list of rank elements
- group elements per article
- build a result tree
- “score grouping”:

- for each element N1
  1. score N2 > score N1
  2. concentration of good elements
  3. even distribution of good elements

Case 1

Case 2

Case 3
CAS query processing: sub-queries

• Sub-queries decomposition
  – //article [search engines] // sec [Internet growth] AND sec [Yahoo]
    • article [search engines]
    • sec [Internet growth]
    • sec [Yahoo]

• Run each sub-queries and then combine
• Reward structure matching (strict vs vague)

(Sauvagnat etal, INEX 2005)
Example of combination: Probabilistic algebra

$R(\text{learning structure}) \cap \text{label}^{-1}(\text{sec})$

$\Rightarrow \text{descendants}(R(\text{bayesian networks}) \cap \text{label}^{-1}(\text{article}))$

- “Vague” sets
  - $R(\ldots)$ defines a vague set of elements
  - $\text{label}^{-1}(\ldots)$ can be defined for strict or vague interpretation
- Intersections and Unions are computed as probabilistic “and” and fuzzy-OR.
Vague structural constraints

- Define score between two tags/paths
- Boost content score with tag/path score
- Use of dictionary of equivalent tags/synonym list
  - Analysis of the collection DTD
    - Syntactic, e.g. “p” and “ip1”
    - Semantic, e.g. “capital” and “city”
  - Analysis of past relevance assessments
    - For topic on “section” element, all types of elements assessed relevant added to “section” synonym list
- Ignore structural constraint for target, support element or both
- Relaxation techniques from DB (e.g. lowest common ancestor, etc)
Conclusions

- Choice of retrieval units can affect the “type” of retrieval models
- XML retrieval can be viewed as a combination of evidence problem
- No “clear winner” in terms of retrieval models
  - We still miss the benchmark/baseline approach
  - Lots of heuristics
- BUT WHAT SEEM TO WORK WELL:
  - Element
  - Document
  - Size

- Thorough investigation for all ranking models, all indexing approaches, and all evidence needed
Part III: Evaluation of XML Retrieval
Evaluation of XML retrieval: INEX

• Evaluating the effectiveness of content-oriented XML retrieval approaches

• Collaborative effort ⇒ participants contribute to the development of the collection
  queries
  relevance assessments
  methodology

• Similar methodology as for TREC, but adapted to XML retrieval

http://inex.is.informatik.uni-duisburg.de/
## Document collections

<table>
<thead>
<tr>
<th>Year</th>
<th>number documents</th>
<th>number elements</th>
<th>size</th>
<th>average number elements</th>
<th>average element depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>2002-2004</td>
<td>12,107</td>
<td>8M</td>
<td>494MB</td>
<td>1,532</td>
<td>6.9</td>
</tr>
<tr>
<td>2005</td>
<td>16,819</td>
<td>11M</td>
<td>764MB</td>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>2006-7</td>
<td>659,388</td>
<td>52M</td>
<td>60(4.6)GB</td>
<td>161.35</td>
<td>6.72</td>
</tr>
</tbody>
</table>

(IEEE, Wikipedia (Denoyer & Gallinari, SIGIR Forum, June 2006))
Two types of topics

- **Content-only (CO) topics**
  - ignore document structure
  - simulates users, who do not have any knowledge of the document structure or who choose not to use such knowledge

- **Content-and-structure (CAS) topics**
  - contain conditions referring both to content and structure of the sought elements
  - simulate users who do have some knowledge of the structure of the searched collection
CO topics 2003-2004

<title>
"Information Exchange", +"XML", "Information Integration"
</title>

<description>
How to use XML to solve the information exchange (information integration) problem, especially in heterogeneous data sources?
</description>

<narrative>
Relevant documents/components must talk about techniques of using XML to solve information exchange (information integration) among heterogeneous data sources where the structures of participating data sources are different although they might use the same ontologies about the same content.
</narrative>
Automated vehicle applications in articles from 1999 or 2000 about intelligent transportation systems.

To be relevant, the target component must be from an article on intelligent transportation systems published in 1999 or 2000 and must include a section which discusses automated vehicle applications, proposed or implemented, in an intelligent transportation system.
Relevance in XML retrieval

- A document is relevant if it “has significant and demonstrable bearing on the matter at hand”.

- Common assumptions in laboratory experimentation:
  - Objectivity
  - Topicality
  - Binary nature
  - Independence

(Borlund, JASIST 2003)
(Goevert et al, JIR 2006)
Relevance in XML retrieval: INEX 2003 - 2004

- Relevance = (0,0) (1,1) (1,2) (1,3) (2,1) (2,2) (2,3) (3,1) (3,2) (3,3)
  - Exhaustivity = how much the section discusses the query: 0, 1, 2, 3
  - Specificity = how focused the section is on the query: 0, 1, 2, 3
- If a subsection is relevant so must be its enclosing section, ...

- Topicality not enough
- Binary nature not enough
- Independence is wrong

(based on Chiaramella etal, FERMI fetch and browse model 1996)
Relevance - to recap

• find smallest component (→ specificity) that is highly relevant (→ exhaustivity)

• specificity: extent to which a document component is focused on the information need, while being an informative unit.

• exhaustivity: extent to which the information contained in a document component satisfies the information need.
Specificity dimension 2005 - continuous scale defined as ratio (in characters) of the highlighted text to element size.

<<TESTING DILEMMA>>

The dilemma in testing embedded DRAM arises from differences in character between ASICs and commodity DRAMs. In the case of commodity DRAMs, despite huge amounts of production, manufacturers produce only a few different products at the same time. As a result, they can optimize the testing methodology for each product. In contrast, companies produce a large variety of ASIC products, but the production volume of each product is small. Also, ASICs require a very short turnaround time. Therefore, customizing the test methodology for each product is difficult. ASICs require a common test environment that covers all product variations.

Furthermore, since the commodity DRAM is a general-purpose product, we cannot specify its application during testing. Thus, testing must cover various kinds of applications and provide very a result, the commodity DRAM's test time is longer than the ASIC's.
Exhaustivity dimension

Scale reduced to 3+1:

- Highly exhaustive (2): the element discussed most or all aspects of the query.
- Partly exhaustive (1): the element discussed only few aspects of the query.
- Not exhaustive (0): the element did not discuss the query.
- Too Small (?): the element contains relevant material but is too small to be relevant on its own.

New assessment procedure led to better quality assessments
(Piwowarski et al, 2007)
HiXEval - Generalized precision and recall based on amount of highlighted content

Ali Baba

Wikipedia link: Ali Baba

Ali Baba

0 conversion warning(s)

Ali Baba (Arabic: علي بابا) is a fictional character described in the adventure tale of "Ali Baba and the Forty Thieves" which was added to the traditional collection of The Book of One Thousand and One Nights by its European transcriber, Antoine Galland, an 18th-century French orientalist who had heard it in oral form from a Maronite story-teller from Aleppo. This story has also been used as a popular pantomime plot.

Story Summary

Ali Baba, a poor woodcutter, happens to see and overhear a large band of thieves - forty in all - visiting their treasure store in the forest where he is cutting wood. The thieves' treasure is in a cave, the mouth of which is sealed by magic - it opens on the words "Open, Sesame", and seals itself on the words "Close, Sesame". When the thieves are gone, Ali Baba enters the cave himself, and takes some of the treasure home.

Ali Baba's rich brother, Kassim, finds out about his brother's unexpected wealth, and Ali Baba tells cave to take more of the treasure, but forgets the magic

For each element, we derive:
  rsize: number of highlighted characters
  size: number of characters

For each topic, we derive:
  Trel: number of highlighted characters in collection
Part IV: Views on XML Retrieval
Views on XML

Content 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

```xml
  
  <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

/document/chapter[about(./heading, XML) AND about(./section//*, syntax)]
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?
Example query:

```
//chapter[about (.,
   XML query language]
```
XML content typing: 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
XML content typing: 3. Object Types

- Object types: Persons, Locations. Companies, ..... 

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 had Picasso 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
DAML+OIL for semantic XML IR?
Conclusion

- XML retrieval as extension of classical text retrieval -> focused retrieval (content-only)
- Considering structure: Various interpretations of Content-and-Structure
- Interactive retrieval: focus+context, usability issues
- XML text retrieval with data/object types still at its infancy