XML/XQuery and Information Retrieval

Thijs Westerveld
Outline

• Information Retrieval
• XML Retrieval
  – IR and structure
  – XQuery full-text
  – INEX/NEXI
• Tijah
  – Layered XML IR system
  – Score region algebra
• PF/Tijah
  – Integration of Tijah and Pathfinder
  – Mixing IR and DB type of queries
Information retrieval
Data vs. Information retrieval

Data retrieval
- Well defined, structured
- Deterministic
- Exact match
- Structured
- Any match
- Arbitrary

Information Retrieval
- Ambiguous, Natural language
- Probabilistic
- Partial or ‘best’ match
- Keywords, Natural language
- Relevance
- Ranked

Retrieval models
- Matching/Retrieval
- Query types
- Results criteria
- Results ordering
IR Terminology

• Document
  – Unit to retrieve
• Collection
  – Set of documents that is searched
• Term
  – Word/lexical item that occurs in collection
• Query
  – Information need expressed as a set of terms
• Full text search
  – Document text instead of manually assigned keywords
• Ad Hoc retrieval
  – Typical IR task: Fixed collection, any query, ranked results
IR models

• Bag of words
  – “I see what I eat” = “I eat what I see”

• Boolean retrieval vs Ranked retrieval
  – Does the term occur vs. to what extent is the document *about* the term

• Ranked retrieval models
  – Vector space model, Probabilistic model, Bayesian network models, Language model

• Term weighting, tf.idf
  – Term frequency
    • How often does a term occur in a document?
  – Inverse document frequency
    • In how many documents does the term

• Link analysis
XML retrieval

• Documents
  – In traditional IR, the retrieval unit is fixed. Often, it’s full documents (or webpages), sometimes it’s paragraphs (or complete websites).
  – In XML IR the retrieval unit can be any of these. It’s up to the system to decide the appropriate level of granularity for a given query.

• Structure
  – The structural relationships in the documents may be exploited to better locate the relevant information.
    • E.g., If a section title is ‘about the query’, it may be better to return the corresponding section than the title itself.
Two XML IR initiatives

- **XQuery full-text**
  - W3C working draft
    - First version July 2004, latest version May 2006
    - Extends XQuery/Xpath with full-text search capabilities
- **INEX**
  - INitiative for the Evaluation of XML retrieval (2002-present)
  - Evaluating the effectiveness of XML IR systems
  - Funded by DELOS Network of excellence for digital libraries
Full-text search vs substring match

[From the Xquery full-text working draft]

- Tokens or phrases rather than substrings.
  - Substring search for lease will return "Foobar Corporation releases the 20.9 version ...", fulltext search will not
- Full-text search will support language-based searches which substring search cannot.
  - Stemming: mouse/mice
  - Proximity: XML and Query within 3 tokens distance
- Full-text search must address the vagaries and nuances of language.
  - Ranked retrieval
  - Searching for mouse may return documents of with "mice", and possibly "rodents", or possibly "computers". Some results are more "mousey" than others. We generally expect to see the most relevant results at the top of the results list.
- As XQuery and XPath evolve, they may apply the notion of score to querying structured data (and follow the Xquery full-text syntax)
for $b score $s
  in /books/book[content ftcontains "web site" && "usability"]
where $s > 0.5
order by $s descending
return <result>
  <title> {$b//title} </title>
  <score> {$s} </score>
</result>
INEX

• Documents
  – 16,819 Articles from IEEE journals (764Mb), 2002-2005
• Topics constructed by participants
  – Content only queries: keywords
  – Content and structure queries: keywords and structural hints/constraints
• Assessments (‘ground truth’) for topic-node pairs
  – Community effort of participants
  – Pooling: only assess results top retrieved documents
  – Assumption:
    • not assessed = not relevant
    • Not completely true, but shown to not influence relative ranking of systems
• Metrics to compare search results across systems
NEXI

- Narrow Extended Xpath for INEX
  - The query language used for INEX
- Subset of Xpath
  - Only descendant steps
- Extended with about filters

  - //section[about(.,navigation)]
  - //article[about(.,//section,navigation)]
  - //article[about(.,transport)]//section[about(., navigation)]
TIJAH

- XML Text retrieval system
- Developed by Utwente/CWI
- Runs on top of MonetDB
- Layered design
  - Conceptual level: NEXI
  - Logical level: Score Region Algebra
  - Physical: MIL
- Parameterizable and adaptable
  - Allows search engine developer to adapt at different levels
  - But also, out of the box solutions for common tasks
Document and data model

//sec[about(.//p, perfect bouquet)]//fig

```
<root>0
  <article>1
    <title>2 Beautiful3 flowers4 </title>5
    ...
  <sec>1111
    ...
    <p>1122
      ... perfect1130 bouquet1131 ...
    </p>1155
    ...
    <fig>1166 ... </fig>1177
    ...
  </sec>1199
  </article>1200
  ...
</root>9999
```
Document and data model

```
//sec[about(.//p, perfect bouquet)]//fig
```

```
<table>
<thead>
<tr>
<th>start</th>
<th>end</th>
<th>name</th>
<th>type</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>9999</td>
<td>root</td>
<td>node</td>
<td>1.000</td>
</tr>
<tr>
<td>1111</td>
<td>1199</td>
<td>sec</td>
<td>node</td>
<td>1.000</td>
</tr>
<tr>
<td>1122</td>
<td>1155</td>
<td>p</td>
<td>node</td>
<td></td>
</tr>
<tr>
<td>1130</td>
<td>1130</td>
<td>perfect</td>
<td>term</td>
<td></td>
</tr>
<tr>
<td>1131</td>
<td>1131</td>
<td>bouquet</td>
<td>term</td>
<td></td>
</tr>
</tbody>
</table>

```

```
<root>0
  <article>1
    <title>2 Beautiful3 flowers4 </title>5
    ...
  </article>1
  <sec>1111
    ...
    <p>1122
      ... perfect1130 bouquet1131 ...
    </p>1155
    ...
    <fig>1166 ... </fig>1177
    ...
  </sec>1199
</article>1200
...
</root>9999
```
XML IR aspects

//sec[about(./p, perfect bouquet)]//fig

- **Element and term selection**
  - \{section, p, fig\}
  - \{perfect\}, \{bouquet\}

- **Element score computation**
  - about(./p, perfect)
  - about(./p, bouquet)

- **Element score combination**
  - about(./p, perfect) && about(./p, bouquet)

- **Element score propagation**
  - //p => //sec
  - //sec => //fig
Score Region Algebra (SRA)

\[
\begin{align*}
\text{sec} &= \sigma_{n='sec', t=node}(C) = \{\ldots,(1111,1199),\ldots\} \\
\text{p} &= \sigma_{n='p', t=node}(C) = \{\ldots,(1122,1155),\ldots\} \\
\text{perfect} &= \sigma_{n='perfect', t=term}(C) = \{\ldots,(1130,1130),\ldots\} \\
\text{bouquet} &= \sigma_{n='bouquet', t=term}(C) = \{\ldots,(1131,1131),\ldots\} \\
\text{fig} &= \sigma_{n='fig', t=node}(C) = \{\ldots,(1166,1177),\ldots\}
\end{align*}
\]
Results

SRA output

\{1166, 1177, fig, node, 0.034\}
\{3322, 3377, fig, node, 0.031\}
\{7733, 7799, fig, node, 0.027\}
\{2211, 2266, fig, node, 0.023\}
## Pathfinder vs. Tijah

### Pathfinder (MonetDB/XQuery)

- **Language:** XQuery. Precise structural querying and XML generation
- **Output:** XML
- **Data Model:** pre/size encoding of nodes. Text-nodes are maintained as single strings
- **Architecture:** Layered query processing generating MIL. Execution on MonetDB

### Tijah

- **Language:** NEXI. Structure and content ranking
- **Output:** Ranked sequence of scored nodes/regions
- **Data Model:** Region model with start/end encoding of nodes and terms
- **Architecture:** Layered query processing generating MIL. Execution on MonetDB
PF/Tijah

• The best of both worlds
  – Retrieving arbitrary parts of textual data.
    • Journal, article, section, paragraph, etc.
    • No up front notion of document.
  – Complex scoring of NEXI queries
    • Score combination
    • Score propagation
  – Flexible results presentation
    • Further XPath/XQuery processing of results
    • Easy access to e.g., title, date, authors
  – Text search combined with database querying
    • E.g., employees from financial department that also worked for sales and sent an email about “tax refunds”
PF/Tijah

- Pathfinder module with full-text search functionality based on the Tijah system
- Design constraints
  - Keep pathfinder untouched
  - Use existing PF functionality where possible
  - Flexibility; support generalized IR models
  - Enable fast retrieval
  - Minimize redundant storage of data
  - Use existing Tijah functionality where possible
PF/Tijah

• Pathfinder and Tijah use similar data models
  – pre/size/level for Pathfinder (can be translated to pre/post)
  – start/end for Tijah

• But, text search cannot run directly on pathfinder index
  – Pathfinder sees text node as single unit
  – Tijah needs to index terms

• PF-light index
  – Light version of Pathfinder index
  – Next to existing Pathfinder index
  – Pre/post for nodes and terms
  – Inverted indexes for fast access
  – No replication of full document content (attributes, processing instructions, etc)
Putting things together

• From Pathfinder
  – Document shredder
    • Speed, loading from URL, caching, DTD processing
  – XML serializer
    • Used to create the PF/Tijah index
  – Loop-lifted descendant step
    • For node-term containment relations

• From Tijah
  – The Score Region Algebra
    • New MIL implementations for SRA functions
  – The NEXI query processor
    • NEXI to SRA translation and SRA to MIL translation
Embedding NEXI in XQuery

- How to call text-ranking within XQuery?
  - Xquery Full-text extends language considerably
  - NEXI proved to be useful for content and structure queries and is used in INEX
  - Text ranking has to fit in XQuery; we want it to be fully compositional with other XQuery expressions
  - Special functions for NEXI queries

- How to return nodes and scores?
  - Simple first-order functions cannot return nodes and scores at the same time
Embedding NEXI in XQuery (cont)

• Set of three functions
  – \texttt{tijah-query-id( node-seq, “NEXIquery”)}
    • Returning a query identifier only
  – \texttt{tijah-nodes(query-id)}
    • returns a ranked list of nodes
  – \texttt{tijah-score(query-id, node)}
    • returns the score of that node

• And a shortcut
  – \texttt{tijah-query(node-seq,”NEXIquery”)}
    • equals to
      \texttt{tijah-nodes(tijah-query-id( node-seq, “NEXIquery”))}
Embedding NEXI in XQuery (cont)

• Example

let $col := doc("mycollection.xml")
let $query_id :=
    tijah-query-id($col,"//sec[about(.//p,perfect bouquet)]//fig"
for $node at $rank in tijah-nodes($query_id)
let $score := tijah-score($query-id,$n)
return
    <item rank=""{$rank}" score=""{$score}"">$
    {$node}
</item>
Parameterization

- **TijahOptions node**

  ```
  let $opt := <TijahOptions
    collection="wikipedia"
    txtmodel_model="LMS"
    returnNumber="10"
    txtmodel_returnall="true"
  />

  let $col := doc("mycollection.xml")
  for $res in
    tijah-query($opt, $col,"//sec[about(.,scoring)]
  return $res//@title
  ```
Adaptation

- Adding a new retrieval model does not affect the logical (SRA) query plans
- All that is needed is a new implementation of the element score computation functions
  - Now: Reimplement MIL function
  - Future: Using declarative language RAM
Mixing IR and XML DB queries

- Database selection to set context + IR query for topic + database query for result format
- E.g., From a database with research papers give me the titles of articles Stefan Manegold wrote about XQuery

```xml
let $m := doc("mycol.xml")//DOC[author="Manegold"]
for $res in tijah-query($m, "//*[about(.,xquery)]")
return $res/title/text()
```
Translating between indexes

• A sequence of nodes created by XQuery expression can be a starting point for NEXI query
  – `tijah-query($m, "//*[about(.,xquery)]")`

• A tijah-nodes result sequence needs to be translated back to the Pathfinder index for output generation or further processing
  – `return $res/title/text()`

• To facilitate this, a table is kept to link pf/tijah-light pre identifiers and XQuery pre’s
  – only for nodes, terms have no XQuery pre
Known Issues

• NEXI query is a string and thus a black box for XQuery
  – no variable usage, syntax checking, type checking
• NEXI and XQuery share some expressiveness

```xml
let $c := doc("myCollection.xml")
return tijah-query($c//DOC//SEC,"//P[about(.,XML)]
```

```xml
let $c := doc("myCollection.xml")
return tijah-query($c,"//DOC//SEC//P[about(.,XML)]
```

• Shift all axes steps to XQuery?
  – problematic for score propagation combination
    • //sec[about(.,x)]//p[about(.,y)]

• Tighter integration?
Summary

• Information retrieval (IR) is less structured than data retrieval
  – Ranking vs Exact matching
• XML IR is retrieval from structured collections
  – Retrieval unit not pre-defined
  – Exploit structure to get to relevant stuff
  – Initiatives: XQuery full-text and INEX/NEXI
• Tijah is a parameterizable XML IR system
  – conceptual (nexi), logical (sra), physical (mil) layers
• PF/Tijah combines best of Pathfinder and Tijah
  – Mixing IR and DB type of queries
  – Translate between PF node-index and PF/Tijah node/term index