Accessing XML Content From DB and IR Perspectives

Sihem Amer-Yahia
AT&T Labs – Research
Mounia Lalmas
Queen Mary University of London

Outline

• Part I: Motivation
  – Information retrieval (IR) vs. Structured Document Retrieval (SDR)
  – XML Search vs. Traditional Search
• Part II: XML Search and Scoring
• Part III: Evaluation
• Part IV: Open Issues
• Part V: Contact Details

Structured Document Retrieval (SDR)

• 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.

• Structure improves precision
• Exploit visual memory
Conceptual Model for IR

- Documents
  - Indexing
  - Document representation
  - Retrieval function
  - Retrieval results

- Query
  - Formulation
  - Query representation
  - Retrieval function
  - Retrieval results

Conceptual Model for SDR

- Structured documents
  - Indexing
  - Document representation
  - Inverted file + structure index
  - Retrieval function
  - Retrieval results

- Content + structure

Outline

- Part I: Motivation
  - Information retrieval (IR) vs. Structured Document Retrieval (SDR)
  - XML Search vs. Traditional Search
- Part II: XML Search and Scoring
- Part III: Evaluation
- Part IV: Open Issues
- Part V: Contact Details

XML is the currently adopted format for structured documents. e.g.XIBQL, NEXI, XQUERY

structure index captures in which document component the term occurs (e.g. title, section), as well as the type of document components (e.g. XML tags)

additional constraints are imposed from the structure

Matching content + structure

e.g. a chapter and its sections may be retrieved

Presentation of related components

Van Rijsbergen 1979
XML: eXtensible Mark-up Language

- Meta-language (user-defined tags) adopted as the document format language by W3C
- Used to describe **content and structure** (and not layout)
- Grammar described in DTD (for validation)

```
<lecture>
  <title> Structured Document Retrieval </title>
  <author> <fnm> Smith </fnm> <snm> John </snm> </author>
  <chapter>
    <title> Introduction into SDR </title>
    <paragraph> … </paragraph>
  </chapter>
</lecture>
```

Use of XPath notation to refer to the XML structure

```
chapter/title: title is a direct sub-component of chapter
chapter/title: title is a direct or indirect sub-component of chapter
chapter/paragraph[2]: any direct second paragraph of any chapter
chapter/*: all direct sub-components of a chapter
```

XML Search

**Motivation**

- XML is able to represent a mix of structured and text (unstructured) information.
- XML applications: *digital libraries, content management*.

**DB and IR views**

- Data-centric view (Database (DB))
  - XML as exchange format for structured data
- Document-centric view (IR)
  - XML as format for representing the logical structure of documents
- Now increasingly both views (DB+IR)
Data Centric XML Documents

Example

```xml
<class name="DCS317" num_of_std="100">
  <lecturer lecid="111">Thomas</lecturer>
  <student marks="70" origin="Oversea">
    <name>Mounia</name>
  </student>
  <student marks="30" origin="EU">
    <name>Tony</name>
  </student>
</class>
```

Document Centric XML Documents

Example

```xml
<class name="DCS317" num_of_std="100">
  <lecturer lecid="111">Mounia</lecturer>
  <student studid="007">
    <name>James Bond</name> is the best student in the class. He scored <interm>95</interm> points out of <max>100</max>. His presentation of <article>Using Materialized Views in Data Warehouse</article> was brilliant.
  </student>
  <student studid="131">
    <name>Donald Duck</name> is not a very good student. He scored <interm>20</interm> points.
  </student>
</class>
```

XML in Library of Congress

http://thomas.loc.gov/home/gpoxmlc109/h2739_ih.xml

```xml
<bill bill-stage="Introduced-in-House">
  <congress>109th CONGRESS</congress> <session>1st Session</session>
  <legis-num>H. R. 2739</legis-num>
  <current-chamber>IN THE HOUSE OF REPRESENTATIVES</current-chamber>
  <action>
    <action-date date="20050526">May 26, 2005</action-date>
    <action-desc>
      <sponsor name-id="T000266">Mr. Tierney</sponsor> (for himself, <cosponsor name-id="M001143">Ms. McCollum of Minnesota</cosponsor>, <cosponsor name-id="M000725">Mr. George Miller of California</cosponsor>) introduced the following bill, which was referred to the <committee-name committee-id="HED00">Committee on Education and the Workforce</committee-name>.
    </action-desc>
  </action>
</bill>
```

THOMAS: Library of Congress

http://thomas.loc.gov/home/gpoxmlc109/h2739_ih.xml
INEX Data
http://inex.is.informatik.uni-duisburg.de/

Example INEX Query

XML Search vs Traditional Search
- **Search Context**: pre-defined XML nodes as a search context.
  - XPath/XQuery queries to specify elements to search.
- **Search Result**: pre-defined meaningful XML fragments to return.
  - element construction with XPath/XQuery.
- **Searching**: keyword search.
  - Boolean operator, proximity distance, scoping, thesaurus, stop words, stemming.
- **Ranking**: taking document structure into account.
  - element-based and path-based scoring.
  - element-size into account.

DB and IR Perspectives
- **DB View**
  - **Expressiveness**: sophisticated queries on structure vs. combining queries on structure with complex full-text search conditions.
  - **Efficiency**: indices and evaluation algorithms for efficient queries on structure. Boolean interpretation.
- **IR View**
  - **Expressiveness**: keyword set only vs. path expressions combined with keyword set.
  - **Effectiveness**: ranking algorithms and precision and recall to measure how well the set of results answer the information need.

Example INEX Query

Example INEX Query

XML Search vs Traditional Search
- **Search Context**: pre-defined XML nodes as a search context.
  - XPath/XQuery queries to specify elements to search.
- **Search Result**: pre-defined meaningful XML fragments to return.
  - element construction with XPath/XQuery.
- **Searching**: keyword search.
  - Boolean operator, proximity distance, scoping, thesaurus, stop words, stemming.
- **Ranking**: taking document structure into account.
  - element-based and path-based scoring.
  - element-size into account.

DB and IR Perspectives
- **DB View**
  - **Expressiveness**: sophisticated queries on structure vs. combining queries on structure with complex full-text search conditions.
  - **Efficiency**: indices and evaluation algorithms for efficient queries on structure. Boolean interpretation.
- **IR View**
  - **Expressiveness**: keyword set only vs. path expressions combined with keyword set.
  - **Effectiveness**: ranking algorithms and precision and recall to measure how well the set of results answer the information need.
Outline

• Part I: Motivation
• Part II: XML Search and Scoring
  – Expressiveness: Languages for XML Search
  – Efficiency: Algorithms for XML Search
  – Effectiveness: XML Scoring and Ranking
• Part III: Evaluation
• Part IV: Open Issues
• Part V: Contact Details

Expressiveness: Languages for XML Search

• Keyword search (CO Queries)
  – 
• 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
XIRQL

<workshop date="28 July 2000">
<title> XML and Information Retrieval: A SIGIR 2000 Workshop </title>
<editors> David Carmel, Yoelle Maarek, Aya Soffer </editors>
<precedings>
<paper id="1">
  <title> XQL and Proximal Nodes </title>
  <author> Ricardo Baeza-Yates </author>
  <author> Gonzalo Navarro </author>
  <abstract> We consider the recently proposed language ... </abstract>
</paper>
<paper id="2">
  <title> XML Indexing </title>
  <author> ... </author>
</paper>
</precedings>
</workshop>

Expressiveness: Languages for XML Search

- 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 := ftextcontains "xml" && "db" distance 5

Similar Notion of Results

- Nearest Concept Queries
  - (Schmidt et al, ICDE 2002)
- XSSearch
  - (Xu & Papakonstantinou, SIGMOD 2005)

XSearch

<workshop date="28 July 2000">
<title> XML and Information Retrieval: A SIGIR 2000 Workshop </title>
<editors> David Carmel, Yoelle Maarek, Aya Soffer </editors>
<precedings>
<paper id="1">
  <title> XQL and Proximal Nodes </title>
  <author> Ricardo Baeza-Yates </author>
  <author> Gonzalo Navarro </author>
  <abstract> We consider the recently proposed language ... </abstract>
</paper>
<paper id="2">
  <title> XML Indexing </title>
  <author> ... </author>
</paper>
</precedings>
</workshop>
Expressiveness: Languages for XML Search

- 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

XPath

- fn:contains($e, string) returns true iff $e contains string
  - //section[fn:contains(.//title, "XML Indexing")]

XIRQL

- Weighted extension to XQL (precursor to XPath)

  - /section[0.6 · .//Section "XQL" +
            0.4 · //section ScwS "syntax"]

XXL

- Introduces similarity operator ~

  Select Z
  From http://www.myzoos.edu/zoos.html
  Where zoos.#.zoo As Z and
  Z.animals.(animal)?.specimen as A and
  A.species ~ "lion" and
  A.birthplace.#.country as B and
  A.region ~ B.content

(Fahr & Snelgrove, SIGIR 2001)
(W3C 2005)
(Fahr & Snelgrove, SIGIR 2001)
(Theobald & Weikum, IJIT 2002)
NEXI

- Narrowed Extended XPath I
- INEX Content-and-Structure (CAS) Queries
- Specifically targeted for content-oriented XML search (i.e., “aboutness”)

```xml
//article[about(.//title, apple) and about(.//sec, computer)]
```

(Trotman & Sigurbjornsson, INEX 2004)

Expressiveness: Languages for XML Search

- 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

Schema-Free XQuery

- Meaningful least common ancestor (mlcas)

```xml
for $a in doc("bib.xml")//author
    $b in doc("bib.xml")//title
    $c in doc("bib.xml")//year
where $a/text() = "Mary" and exists mlcas($a,$b,$c)
return <result> {$b,$c} </result>
```

(Li, Yu, Jagadish, VLDB 2003)

XQuery Full-Text

- Two new XQuery constructs
  1) FTContainsExpr
     - Expresses “Boolean” full-text search predicates
     - Seamlessly composes with other XQuery expressions
  2) FTScoreClause
     - Extension to FLWOR expression
     - Can score FTContainsExpr and other expressions

(W3C 2005)
FTContainsExpr

//book ftcontains “Usability” & “testing” distance 5
//book[./content ftcontains “Usability” with stems]/title
//book ftcontains /article[author=“Dawkins”]/title

FTScore Clause

In any FOR $v [SCORE $s]? IN [FUZZY] Expr
order LET ...
   WHERE ...
ORDER BY ...
RETURN

Example
FOR $b SCORE $s in
  /pub/book/. ftcontains “Usability” & “testing”
ORDER BY $s
RETURN <result score={$s}> $b </result>

In any FOR $v [SCORE $s]? IN [FUZZY] Expr
order LET ...
   WHERE ...
ORDER BY ...
RETURN

Example
FOR $b SCORE $s in F UZZY
  /pub/book/. ftcontains “Usability” & “testing”
ORDER BY $s
RETURN $b
**XQuery Full-Text Evolution**

- **2002**: Quark Full-Text Language (Cornell)
- **2003**: TeXQuery (Cornell, AT&T Labs)
- **2004**: XQuery Full-Text
- **2005**: XQuery Full-Text (Next Drafts)

**Outline**

- **Part I: Motivation**
- **Part II: XML Search and Scoring**
  - Expressiveness: Languages for XML Search
  - Efficiency: Algorithms for XML Search
  - Effectiveness: XML Scoring and Ranking
- **Part III: Evaluation**
- **Part IV: Open Issues**
- **Part V: Contact Details**

**Efficiency: Algorithms for XML Search**

- Mainly a DB issue
- 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 $s :=$ btcontains "xml" && "db"
    distance 5

**Efficiency: Main Issue**

- Given: Query keywords
- Compute: Least Common Ancestors (LCAs) that contain query keywords, in ranked order
Efficiency: Naïve Method

**Naïve inverted lists:**
- Ricardo: 1, 5, 6, 8
- XQL: 1, 5, 6, 7

**Problems:**
1. Space Overhead
2. Spurious Results

Main issue: Decouples representation of ancestors and descendants

Dewey Encoding of IDs [1850s]

DIL: Query Processing
- Merge query keyword inverted lists in Dewey ID Order
  - Entries with common prefixes are processed together
- Compute Longest Common Prefix of Dewey IDs during the merge
  - Longest common prefix ensures most specific results
  - Also suppresses spurious results
- Keep top-m results seen so far in output heap
  - Calculate rank using two-dimensional proximity metric
  - Output contents of output heap after scanning inverted lists
- Algorithm works in a single scan over inverted lists

**Store IDs of elements that directly contain keyword**
- Avoids space overhead

**XRank: Dewey Inverted List (DIL)**

- **Dewey Id**
  - Sorted by Dewey Id
  - XQL: 5.0.3.0.0 85 32 8.0.3.8.3 38 89 91
  - Ricardo: 5.0.3.0.1 82 38 8.2.1.4.2 99 52

- **Score**
- **Position List**
XRank: Ranked Dewey Inverted List (RDIL)

RDIL: Algorithm

• An element may be ranked highly in one list and low in another list
  – B+-tree helps search for low ranked element
• When to stop scanning inverted lists?
  – Based on Threshold Algorithm [Fagin et al., 2002], which periodically calculates a threshold
  – Can stop if we have sufficient results above the threshold
  – Extension to most specific results

RDIL: Query Processing

ID Order vs. Rank Order

• Approaches that combine benefits.
  • Long ID inverted list, short score inverted list
    – HDIL (Guo et al., SIGMOD 2003)
  • Chunk inverted list based on score, organize by ID within chunk
    – FlexPath (Amer-Yahia et al., SIGMOD 2004)
    – SVR (Guo et al., ICDE 2005)
Efficiency: *Algorithms for XML Search*

- 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

XSearch Technique

- Given: An interconnection relationship $R$ between nodes (semantic relationship)
  - $R$ is reflexive and symmetric
- Node interconnection index
  - Given two nodes $n$ and $n'$ in a document $d$, find if $(n,n')$ are in $R^*$
- Use dynamic programming to compute closure
  - Online vs. offline

Efficiency: *Algorithms for XML Search*

- Element Path Index (EPI)
  - Evaluates simple path expressions
- Element Content Index (ECI)
  - Traditional inverted list (but replicates nested elements)
- Ontology Index (OI)
  - Lookup similar concepts (for evaluating ~e)
  - Returned in ranked order

XXL Indexing
Integrating Structure and IL

Efficiency: Algorithms for XML Search

- 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 $s := fcontains "xml"$b && fcontains "db"
distance 5

Scoring Functions Critical for Top-κ Query Processing

- Top-κ answer quality depends on scoring function.
- Efficient top-κ query processing requires scoring function to be:
  - Monotone.
  - Fast to compute.

Structural Join Relaxation

(Al-Khalifa et al, ICDE 2003)
Outline

- Part I: Motivation
- Part II: XML Search and Scoring
  - Expressiveness: Languages for XML Search
  - Efficiency: Algorithms for XML Search
  - Effectiveness: XML Scoring and Ranking
- Part III: Evaluation
- Part IV: Open Issues
- Part V: Contact Details

Effectiveness: XML Scoring and Ranking

Mainly an IR issue

- Score value should reflect relevance of answer to user query. Higher scores imply a higher degree of relevance.
- For queries containing conditions on structure, structural conditions may affect scoring.
- Existing proposals extend common scoring methods: e.g. probabilistic or vector-based similarity.

Lots of “scores”

Keywords (term) scores (weights)

<table>
<thead>
<tr>
<th>Article</th>
<th>Title</th>
<th>Section 1</th>
<th>Section 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.9 XML</td>
<td>0.5 XML</td>
<td>0.2 XML</td>
</tr>
<tr>
<td></td>
<td>0.4 search</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

No fixed retrieval unit + nested document components:
- How to obtain document and collection statistics (e.g. tf, idf)
- Inner aggregation or outer aggregation?

Relationship (augmentation) scores

<table>
<thead>
<tr>
<th>Article</th>
<th>Title</th>
<th>Section 1</th>
<th>Section 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.9 XML</td>
<td>0.5 XML</td>
<td>0.2 XML</td>
</tr>
<tr>
<td></td>
<td>0.4 search</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Nested document components:
- Which components contribute best to content of Article?
- How to estimate weights (e.g. size, number of children)?
Different types of document components:
- Which component is a good retrieval unit?
- Is element size an issue?
- How to estimate component scores (frequency, user studies, size)?

Overlapping elements:
- Which component to return to reduce overlap?
- Should the decision be based on user studies, size, types, etc?

Approaches for element scoring:
- Divergence from randomness
- Vector space model
- Cognitive model
- Boolean model
- Logistic regression
- Approximate structure predicates in DB
- Fusion
- Collection statistics
- Proximity search
- Relevance feedback
- Parameter estimation
- Component statistics
- Ontology
- Term statistics
- Natural language processing
- Bayesian network
- Machine learning
- Language model
- Tuning
- Belief model
- Probabilistic model
- Term statistics
- Natural language processing

Effectiveness: XML Scoring and Ranking
- Keyword queries
  - Compute possibly different scores.
- Tag + Keyword queries
  - Compute scores based on tags and keywords.
- Path Expression + Keyword queries
  - Compute scores based on paths and keywords.
- XQuery + Complex full-text queries
  - Compute scores for (newly constructed) XML fragments satisfying XQuery (structural, full-text and scalar conditions).
Vector space model

<table>
<thead>
<tr>
<th>Index Type</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>article index</td>
<td>$RSV$</td>
</tr>
<tr>
<td>abstract index</td>
<td>$RSV$</td>
</tr>
<tr>
<td>section index</td>
<td>$RSV$</td>
</tr>
<tr>
<td>sub-section index</td>
<td>$RSV$</td>
</tr>
<tr>
<td>paragraph index</td>
<td>$RSV$</td>
</tr>
</tbody>
</table>

if idf as for fixed and non-nested retrieval units

(Mass & Mandelbrod, INEX 2003)

XML-IR

- Topic processor
- Filter
- Extractor
- Ranker
- Merger
- Result

- Word Number
- IDF
- Similarity
- Proximity
- TFIDF

(Mass & Mandelbrod, INEX 2003)

Merging - Normalisation

- BM25
- DFR
- SLM
- Ranking

Language model

- Collection language model
- Smoothing parameter $l$

- Element score

- Element size
- Element score
- Article score

- Query expansion with blind feedback
- Ignore elements with $\geq 20$ terms

(Kamp et al, INEX 2003)
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.

(Clarke, SIGIR 2005)

XXL

- Compute similar terms with relevance score $r_1$ using an ontology.
- Compute TFIDF of each term for a given element content with relevance score $r_2$.
- Relevance of an element content for a term is $r_1 \times r_2$.
- $r_1$ and $r_2$ are computed as a weighted distance in an ontology graph.
- Probabilities of conjunctions multiplied (independence assumption) along elements of same path to compute path score.

(Theobald & Weikum, EDBT 2003)

ElemRank

- TFIDF to compute weight of keyword for a leaf element.
- A vector is associated with each non-leaf element.
- $sim(Q,N)$: sum of the cosine distances between the vectors associated with nodes in $N$ and vectors associated with terms matched in $Q$.

(Gue et al, SIGMOD 2003)

XSearch

- Compute similar terms with relevance score $r_1$ using an ontology.
- Compute TFIDF of each term for a given element content with relevance score $r_2$.
- Relevance of an element content for a term is $r_1 \times r_2$.
- $r_1$ and $r_2$ are computed as a weighted distance in an ontology graph.
- Probabilities of conjunctions multiplied (independence assumption) along elements of same path to compute path score.
Effectiveness: XML Scoring and Ranking

- Keyword queries
  - compute possibly different scores.
- Tag +Keyword queries
  - compute scores based on tags and keywords.
- Path Expression + Keyword queries
  - compute scores based on paths and keywords.
- XQuery + Complex full-text queries
  - compute scores for (newly constructed) XML fragments satisfying XQuery (structural, full-text and scalar conditions).

Vector–based Scoring (JuruXML)

- Transform query into (term/path) conditions:
  article/bm/bib/bibl/bb\[about(., hypercube mesh torus nonnumerical database)]
- (term,path)-pairs:
  hypercube, article/bm/bib/bibl/bb
  mesh, article/bm/bib/bibl/bb
torus, article/bm/bib/bibl/bb
nonnumerical, article/bm/bib/bibl/bb
database, article/bm/bib/bibl/bb
- Modified cosine similarity as retrieval function for vague matching of path conditions.

JuruXML Vague Path Matching

Modified vector-based cosine similarity

\[
cr (\text{article/bibl, article/bm/bib/bibl/bb}) = 1/2 + 5/3/6 = 0.5
\]

Probabilistic algebra

\[
R(\text{learning structure}) \sqcap \text{label1}(\text{sec})
\sqcap \text{descendants}(R(\text{"bayesian networks"}) \sqcap \text{label2}(\text{article}))
\]

- “Vague” sets
  - \(R(\ldots)\) defines a vague set of doxels
  - label-1(\ldots) can be defined as strict for SCAS or vague for VCAS
- Intersections and Unions are computed as probabilistic “and” and fuzzy-or.

(Mass et al, INEX 2002)
XIRQL: Probabilistic Scoring

- Extension of XPath.
- Weighting and ranking:
  - weighting of query terms:
    \[ P\text{sum}(0.6,a) = 0.6 \cdot P(a) + 0.4 \cdot P(b) \]
  - probabilistic interpretation of Boolean connectors:
    \[ P(a \& \& b) = P(a) \cdot P(b) \]

(Furh & Großjohann, SIGIR 2001)

XIRQL Example

- Query:
  - “Search for an artist named Ulbrich, living in Frankfurt, Germany about 100 years ago”
- Data:
  - “Ernst Olbrich, Darmstadt, 1899”
- Weights and ranking:
  - \[ P(\text{Olbrich} \ p \ Ulbrich) = 0.8 \] (phonetic similarity)
  - \[ P(1899 \ n \ 1903) = 0.9 \] (numeric similarity)
  - \[ P(\text{Darmstadt} \ g \ Frankfurt) = 0.7 \] (geographic distance)

XML Query Relaxation (FlexPath)

where DB and IR meet

- Tree pattern relaxations:
  - Leaf node deletion
  - Edge generalization
  - Node promotion

A Family of XML Scoring Methods

- Twig scoring
  - High quality
  - Expensive computation
- Path scoring
- Binary scoring
  - Low quality
  - Fast computation

Effectiveness: *XML Scoring and Ranking*

- Keyword queries
  - compute possibly different scores.
- Tag + Keyword queries
  - compute scores based on tags and keywords.
- Path Expression + Keyword queries
  - compute scores based on paths and keywords.
- XQuery + Complex full-text queries
  - compute scores for (newly constructed) XML fragments satisfying XQuery (structural, full-text and scalar conditions).

XIRQL + Relaxation

*where DB and IR should meet*

- XIRQL proposes vague predicates but it is not clear how to combine it with all of XQuery.
- Open issue as how to relax all of XQuery including structured and scalar predicates.

Outline

- Part I: Motivation
- Part II: XML Search and Scoring
- Part III: Evaluation
  - Before INEX and INEX
  - Topics and Tasks
  - Relevance Assessment
  - Metrics
  - User Study
  - INEX 2005
- Part IV: Open Issues
- Part V: Contact Details

Evaluation

- Mainly done with respect to effectiveness:
  - how good are the XML systems at returning the answers that users are looking for.
- Effectiveness:
  - test collection + topics + relevance assessments + metrics
- Major advances in XML search (ranking) approaches made possible with INEX
Passage retrieval
- Test collection built for that purpose, where passages in relevant documents were assessed (Wilkinson, SIGIR 1994)

Structured document retrieval
- Web retrieval collection (museum) (Lalmas & Moutoussari, RIAO 2000)
- Fictitious collection (Rutten et al, 2Dec 1998; Roelleke et al, ECIR 2000)
- Shakespeare collection (Kazai et al, ECIR 2003)

- “Real” large test collection following TREC methodology
- Evaluation campaign
- XML

Evaluation of XML retrieval: INEX
- Evaluating the effectiveness of content-oriented XML retrieval approaches

Collaborative effort by participants contribute to the development of the collection queries
- Relevance assessments

Similar methodology as for TREC, but adapted to XML

INEX test suites
- Documents ~ 500MB (+ 241 MB): 12,107 (16,018) articles in XML format from IEEE Computer Society; 8 millions elements (+ Now Lonely Planet data, 15 MB)

- INEX 2002
  60 topics, inex_eval metric

- INEX 2003
  66 topics, use subset of XPath, inex_eval and inex_eval_ng metrics

- INEX 2004
  75 topics, subset of 2003 XPath subset (NEXI)
  Official metric: inex_eval with averaged different “assumed user behaviours”
  Others: inex_eval_ng, XCG, G2, ERR, PRUM, ...

- INEX 2005
  87 topics, NEXI Retrieval strategy
  Official metric: XCG
INEX Topics: Content-only

<title>Open standards for digital video in distance learning</title>

<description>Open technologies behind media streaming in distance learning projects</description>

<narrative> I am looking for articles/components discussing methodologies of digital video production and distribution that respect free access to media content through internet or via CD-ROMs or DVDs in connection to the learning process. Discussions of open versus proprietary standards of storing and sending digital video will be appreciated. </narrative>

<keywords>media streaming, video streaming, audio streaming, digital video, distance learning, open standards, free access</keywords>

INEX Topics: Content-and-structure

<title>//article[about(., 'formal methods verify correctness aviation systems')]/sec/*[about(., 'case study application model checking theorem proving')]/</title>

<description>Find documents discussing formal methods to verify correctness of aviation systems. From those articles extract parts discussing a case study of using model checking or theorem proving for the verification.</description>

<narrative>To be considered relevant a document must be about using formal methods to verify correctness of aviation systems, such as flight traffic control systems, airplane- or helicopter- parts. From those documents a section-part must be returned (I do not want the whole section, I want something smaller). That part should be about a case study of applying a model checker or a theorem prover to the verification. </narrative>

<keywords>SPIN, SMV, PVS, SPARK, CWB</keywords>

Content-and-structure topics: Restrictions

- Returning "attribute" type elements (e.g. author, date) not allowed
  "return authors of articles containing sections on XML retrieval approaches"

- Aboutness criterion must be specified - at least - in the target elements:
  "return all paragraphs contained in sections that discuss XML retrieval approaches"

- Branches not allowed (up to 2004)
  "return sections about XML retrieval that are contained in articles that contain paragraphs about INEX experiments"

- ...
Outline

- Part I: Motivation
- Part II: XML Search and Scoring
- Part III: Evaluation
  - Before INEX and INEX
  - Topics and Tasks
  - Relevance Assessment
  - Metrics
  - User Study
  - INEX 2005
- Part IV: Open Issues
- Part V: Contact Details

Relevance in information retrieval

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

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

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

Relevance in XML retrieval: INEX 2003 - 2004

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

- Relevance = (0,0) (1,1) (1,2) (1,3) (2,1) (2,2) (2,3) (3,1) (3,2) (3,3)
  - Exclusivity = 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, ...

(based on Chiaramella et al., 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.

Relevance assessment task

- Topics are assessed by the INEX participants
- Pooling technique (~500 elements on runs of 1500 elements)

- Completeness
  - Rules that force assessors to assess related elements
  - E.g. element assessed relevant [ ] its parent element and children elements must also be assessed
  - …

- Consistency
  - Rules to enforce consistent assessments
  - E.g. Parent of a relevant element must also be relevant, although to a different extent
  - E.g. Exhaustivity increases going up; specificity increases going down
  - …

Assessments: some results

- With respect to the elements to assess
  - 26 % assessments on elements in the pool (66 % in INEX 2002).
  - 68 % highly specific elements (3,3) not in the pool

- 7 % elements automatically assessed

- INEX 2002
  - 23 inconsistent assessments per query for one rule
% Agreement per topic

<table>
<thead>
<tr>
<th>Topic</th>
<th>Nonzero agreement(%)</th>
<th>Exact agreement(%)</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12.59</td>
<td>2.01</td>
<td>CAS</td>
</tr>
<tr>
<td>2</td>
<td>2.95</td>
<td>0.21</td>
<td>CAS</td>
</tr>
<tr>
<td>3</td>
<td>22.85</td>
<td>10.01</td>
<td>CAS</td>
</tr>
<tr>
<td>4</td>
<td>8.60</td>
<td>1.79</td>
<td>CAS</td>
</tr>
<tr>
<td>5</td>
<td>60.87</td>
<td>47.83</td>
<td>CAS</td>
</tr>
<tr>
<td>6</td>
<td>0.00</td>
<td>0.00</td>
<td>CAS</td>
</tr>
<tr>
<td>7</td>
<td>27.53</td>
<td>9.66</td>
<td>CAS</td>
</tr>
<tr>
<td>8</td>
<td>7.63</td>
<td>1.55</td>
<td>CO</td>
</tr>
<tr>
<td>9</td>
<td>25.22</td>
<td>6.72</td>
<td>CO</td>
</tr>
<tr>
<td>10</td>
<td>9.89</td>
<td>0.35</td>
<td>CO</td>
</tr>
<tr>
<td>11</td>
<td>5.65</td>
<td>0.76</td>
<td>CO</td>
</tr>
<tr>
<td>12</td>
<td>9.08</td>
<td>0.96</td>
<td>CO</td>
</tr>
</tbody>
</table>

% Agreement CO vs. CAS

% Agreement per tag

<table>
<thead>
<tr>
<th>Tag</th>
<th>Nonzero agreement(%)</th>
<th>Exact agreement(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abs</td>
<td>7.53</td>
<td>3.23</td>
</tr>
<tr>
<td>App</td>
<td>13.64</td>
<td>3.03</td>
</tr>
<tr>
<td>Art</td>
<td>2.44</td>
<td>0.00</td>
</tr>
<tr>
<td>Art/a</td>
<td>21.70</td>
<td>6.93</td>
</tr>
<tr>
<td>As</td>
<td>1.95</td>
<td>0.24</td>
</tr>
<tr>
<td>B</td>
<td>16.45</td>
<td>12.25</td>
</tr>
<tr>
<td>Bb</td>
<td>15.37</td>
<td>14.25</td>
</tr>
<tr>
<td>Bdy</td>
<td>20.33</td>
<td>5.12</td>
</tr>
<tr>
<td>Bib</td>
<td>14.84</td>
<td>1.29</td>
</tr>
<tr>
<td>Bm</td>
<td>15.79</td>
<td>2.63</td>
</tr>
<tr>
<td>Fig</td>
<td>20.25</td>
<td>7.59</td>
</tr>
<tr>
<td>Fm</td>
<td>6.06</td>
<td>1.52</td>
</tr>
<tr>
<td>Index-entry</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Ip1</td>
<td>10.11</td>
<td>2.45</td>
</tr>
<tr>
<td>Item</td>
<td>10.16</td>
<td>2.34</td>
</tr>
<tr>
<td>Lists (sum)</td>
<td>5.14</td>
<td>0.00</td>
</tr>
<tr>
<td>P</td>
<td>9.51</td>
<td>2.09</td>
</tr>
<tr>
<td>P2</td>
<td>10.84</td>
<td>3.20</td>
</tr>
<tr>
<td>Ref</td>
<td>5.00</td>
<td>3.00</td>
</tr>
<tr>
<td>Sec</td>
<td>15.90</td>
<td>3.12</td>
</tr>
<tr>
<td>Ss1</td>
<td>14.01</td>
<td>2.09</td>
</tr>
<tr>
<td>Ss2</td>
<td>10.45</td>
<td>0.75</td>
</tr>
<tr>
<td>St</td>
<td>5.94</td>
<td>0.99</td>
</tr>
</tbody>
</table>

Outline

- Part I: Motivation
- Part II: XML Search and Scoring
- Part III: Evaluation
  - Before INEX and INEX
  - Topics and Tasks
  - Relevance Assessment
  - Metrics
  - User Study
  - INEX 2005
- Part IV: Open Issues
- Part V: Contact Details
Measuring effectiveness: Metrics

• Need to consider:
  - Two four-graded dimensions of relevance
  - Overlapping elements in retrieval runs
  - Overlapping elements in recall-base
  - ...

• Metrics (INEX 2003 workshop, INEX IR festival 2005)
  - inex_eval (also known as inex2002)
  - inex_eval_ng (also known as inex2003)
  - ERR (expected ratio of relevant units)
  - XCG (XML cumulative gain)
  - O (tolerance to irrelevance)
  - PRUM (Precision Recall with User Modelling)

Overlap in XML Retrieval

- Overlapping (nested) result elements retrieval runs
- Overlapping (nested) reference elements in recall-base

Relevance propagates up!

• ~26,000 relevant elements on ~14,000 relevant paths
• Propagated assessments: ~45%
• Increase in size of recall-base: ~182%

Retrieve the best XML elements according to content and structure criteria:

• Most exhaustive and the most specific = (3,3)

• Near misses = (3,3) + (2,3) (1,3) <= specific
• Near misses = (3,3) + (3,2) (3,1) <= exhaustive
• Near misses = (3,3) + (2,3) (1,3) (3,2) (3,1) (1,2) …

• XML retrieval = no overlapping elements
Two four-graded dimensions of relevance

- How to differentiate between (1,3) and (3,3), ...?
- Several “user models”
  - Expert and impatient: only reward retrieval of highly exhaustive and specific elements (3,3)
  - Expert and patient: only reward retrieval of highly specific elements (3,3), (2,3) (1,3)
  - Naïve and has lots of time: reward - to a different extent - the retrieval of any relevant elements; i.e. everything apart (0,0)
- Use a quantisation function for each “user model”

Examples of quantisation functions

**Expert and impatient**

\[
f_{exi}(\text{exh}, \text{spec}) = \begin{cases} 
1 & \text{if } \text{exh} = 3 \text{ and } \text{spec} = 3 \\
0 & \text{otherwise}
\end{cases}
\]

**Naïve and has a lot of time**

\[
f_{nai}(\text{exh}, \text{spec}) = \begin{cases} 
1 & \text{if } (\text{exh}, \text{spec}) = (3,3) \\
0.75 & \text{if } (\text{exh}, \text{spec}) = ([2,3],[1,2],[3,1]) \\
0.50 & \text{if } (\text{exh}, \text{spec}) = ([1,3],[2,2],[2,1]) \\
0.25 & \text{if } (\text{exh}, \text{spec}) = ([1,1],[1,2]) \\
0.00 & \text{if } (\text{exh}, \text{spec}) = (0,0)
\end{cases}
\]

inex_eval

- Based on precall (Raghavan et al., TOIS 1989) itself based on expected search length (Cooper, JASIS 1968)
- Use several quantisation functions
- Overall performance as simple average across quantisation functions
- In its form cannot consider
  - overlap in retrieval runs
  - overlap in recall-base
- Not easy way to extend to consider BOTH

Overlap in results

**Simulated runs**

Figure 9. Greenish: precom-recall. The sets of alphanum representations recall and the sets of continuous the precom. Precision are averaged over the system.
Overlap in results

Official INEX 2004 Results for CO topics

<table>
<thead>
<tr>
<th>Rank</th>
<th>Systems (runs)</th>
<th>Avg Prec</th>
<th>% Overlap</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>IBM Haifa Research Lab (CO-0.5-LAREFINEMENT)</td>
<td>0.1437</td>
<td>80.89</td>
</tr>
<tr>
<td>2.</td>
<td>IBM Haifa Research Lab (CO-0.5)</td>
<td>0.1346</td>
<td>81.64</td>
</tr>
<tr>
<td>3.</td>
<td>University of Waterloo (Waterloo-Baseline)</td>
<td>0.1276</td>
<td>76.32</td>
</tr>
<tr>
<td>4.</td>
<td>University of Amsterdam (UAms-CO-T-FBack)</td>
<td>0.1274</td>
<td>81.35</td>
</tr>
<tr>
<td>5.</td>
<td>University of Waterloo (Waterloo-Expanded)</td>
<td>0.1173</td>
<td>75.62</td>
</tr>
<tr>
<td>6.</td>
<td>Queensland University of Technology (CO_PS_099_049)</td>
<td>0.1073</td>
<td>79.04</td>
</tr>
<tr>
<td>7.</td>
<td>Queensland University of Technology (CO_PS_099_049)</td>
<td>0.1072</td>
<td>76.81</td>
</tr>
<tr>
<td>8.</td>
<td>IBM Haifa Research Lab (CO-0.5-Clustering)</td>
<td>0.1043</td>
<td>81.10</td>
</tr>
<tr>
<td>9.</td>
<td>University of Amsterdam (UAms-CO-T)</td>
<td>0.1030</td>
<td>71.96</td>
</tr>
<tr>
<td>10.</td>
<td>LIR (simple)</td>
<td>0.0923</td>
<td>64.29</td>
</tr>
</tbody>
</table>

Effect of quantisations

<table>
<thead>
<tr>
<th>Linear correlation coefficient</th>
<th>Spearman rank coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>strict – generalised</td>
<td>0.89</td>
</tr>
<tr>
<td>strict-spec</td>
<td>0.84</td>
</tr>
<tr>
<td>strict-e3321</td>
<td>0.97</td>
</tr>
<tr>
<td>strict-e321x3</td>
<td>0.8</td>
</tr>
<tr>
<td>generalised-spec</td>
<td>0.99</td>
</tr>
<tr>
<td>generalised-e3321</td>
<td>0.87</td>
</tr>
<tr>
<td>generalised-e321x3</td>
<td>0.97</td>
</tr>
</tbody>
</table>

XCG: XML cumulated gain

- Based on cumulated gain measure for IR (Kekäläinen and Järvelin, TOIS 2002)
- Accumulate gain obtained by retrieving elements up to a given rank; thus not based on precision and recall
- Require the construction of an ideal recall-base and associated ideal run, with which retrieval runs are compared
- Consider overlap in both retrieval runs and recall-base

(Kazai et al SIGIR 2004, INEX 2004)
Cumulated Gain

- Gain vector (G) from ranked document list
- Ideal gain vector (I) from documents in recall-base
- Cumulated gain (CG)

\[ CG[j] = \sum_{i} G[i] \]

\[ CG = \langle 3, 3, 4, 7, 9 \rangle \]
\[ I = \langle 3, 3, 2, 1, 0 \rangle \]
\[ G = \langle 3, 0, 1, 3, 2 \rangle \]

Ideal Recall-base and Run

- Ideal recall-base - which non-overlapping elements do we keep?
  - Derived based on retrieval tasks (user models)

- Ideal run - how do we order the above elements?
  - Ordering elements of the ideal recall-base by relevance value \( rv \)

XCG - Ideal run vs. actual run

Recall-base:

- Ideal gain vector
  \[ I[i] = rv(c_i) \]
  \( rv(c_i) \) from ideal recall-base

Ranked result list:

- Actual gain vector
  \[ G[i] = rv(c_i) \]
  \( rv(c_i) \) from full recall-base

XCG: Top 10 INEX 2004 runs

<table>
<thead>
<tr>
<th>Team/CG</th>
<th>overlap</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIP6 (bn-m1-eqt-porder-eul-parameters-00700) [27]</td>
<td>0.3247</td>
<td>74.31</td>
</tr>
<tr>
<td>U. of Tampere (UTampere CO fuzzy) [42]</td>
<td>0.07</td>
<td>50.22</td>
</tr>
<tr>
<td>LIP6 (bn-m1-eqt-porder-eul-parameters-00195) [27]</td>
<td>0.3247</td>
<td>74.31</td>
</tr>
<tr>
<td>U. of Tampere (UTampere CO overlap) [25]</td>
<td>0.3247</td>
<td>74.31</td>
</tr>
<tr>
<td>U. of Tampere (UTampere CO average) [40]</td>
<td>0.3247</td>
<td>74.31</td>
</tr>
<tr>
<td>U. of Tampere (UTampere CO FUS) [23]</td>
<td>0.3247</td>
<td>74.31</td>
</tr>
<tr>
<td>U. of Tampere (UTampere CO overlap) [25]</td>
<td>0.3247</td>
<td>74.31</td>
</tr>
<tr>
<td>LIP6 (bn-m1-eqt-porder-eul-parameters-00195) [27]</td>
<td>0.3247</td>
<td>74.31</td>
</tr>
<tr>
<td>U. of Tampere (UTampere CO overlap) [25]</td>
<td>0.3247</td>
<td>74.31</td>
</tr>
<tr>
<td>LIP6 (bn-m1-eqt-porder-eul-parameters-00700) [27]</td>
<td>0.3247</td>
<td>74.31</td>
</tr>
</tbody>
</table>

[7] rank by inex_eval
Outline

• Part I: Motivation
• Part II: XML Search and Scoring
• Part III: Evaluation
  – Before INEX and INEX
    – Topics and Tasks
    – Relevance Assessment
    – Metrics
    – User Study
    – INEX 2005
• Part IV: Open Issues
• Part V: Contact Details

Interactive Track in 2004

• Investigate behaviour of searchers when interacting with XML components
• Content-only Topics
  – topic type an additional source of context
  – Background topics / Comparison topics
  – 2 topic types, 2 topics per type
  – 2004 INEX topics have added task information
• Searchers
  – “distributed” design, with searchers spread across participating sites

Topic Example

<title>new +Fortran +90 +compilers<title>
<description>How does a Fortran 90 compiler differ from a compiler for the Fortran before it?</description>
<narrative>I've been asked to make my Fortran compiler compatible with Fortran 90 so I'm interested in the features Fortran 90 added to the Fortran standard before it. I'd like to know about computers they would have been new when they were introduced, especially compilers whose source code might be available. Discussion of people's experience with these features when they were new to them is also relevant. An element will be judged as relevant if it discusses features that Fortran 90 added to Fortran.</narrative>
<keywords>new Fortran 90 compiler</keywords>

Baseline system
Baseline system

Some results

- How far down the ranked list?
  - 83% from rank 1-10
  - 10% from rank 11-20

- Query operators rarely used
  - 80% of queries consisted of 2, 3, or 4 words

- Accessing components
  - ~2/3 was from the ranked list
  - ~1/3 was from the document structure (ToC)

- 1st viewed component from the ranked list
  - 40% article level, 36% section level, 22% ss1 level, 4% ss2 level

- ~70% only accessed 1 component per document

Outline

- Part I: Motivation
- Part II: XML Search and Scoring
- Part III: Evaluation
  - Before INEX and INEX
  - Topics and Tasks
  - Relevance Assessment
  - Metrics
  - User Study
  - INEX 2005
- Part IV: Open Issues
- Part V: Contact Details

INEX 2005

- New tasks and retrieval strategies
- Relevance assessments procedure simplified!
- XCG adopted as the official metric
- More documents added from IEEE CS (up to mid-year 2004)

- Tracks
  - interactive, heterogeneous, mining, relevance feedback, natural language, multimedia
INEX 2005 - Ad hoc tasks

- CO sub-task = content-oriented XML retrieval using content-only conditions
- +S sub-task = content-oriented XML retrieval using additional structural hints
  - Focused retrieval strategy: find the most relevant element in a path
  - Thorough retrieval strategy: find all highly relevant elements
  - Fetch and browse retrieval strategy: first identify relevant articles then identify most relevant specific elements within these articles

- CAS sub-task = content-oriented XML retrieval based on content-and-structure queries
  - VVCAS strategy: structural constraints in target support elements interpreted as vague.
  - SVCAS strategy: structural constraints target elements interpreted as strict and structural constraints in support elements interpreted as vague.
  - VSCAS strategy: structural constraints in target elements interpreted as vague and structural constraints in support elements interpreted as strict.
  - SSCAS strategy: structural constraints in target and support elements interpreted as strict.

INEX 2005 - Relevance assessments

- First pass:
  - Assessors highlight text fragments containing only relevant information.
  - Specificity calculated automatically as some ratio

- Second pass:
  - Assessors judge exhaustivity level of any XML elements having highlighted parts
  - Three levels: highly, partially, not exhaustive

INEX 2005 - Metrics

- Official XCG
  - Variants of XCG metric
  - XCG value at cut-off value
  - Recall-based version of XCG
  - Overlap %

- Others:
  - inex_eval
  - inex_eval_ng
  - PRUM
  - ERR
  - trec_eval
Open Issues (in no particular order)

- Difficult research issues in XML retrieval are not ‘just’ about the effective retrieval of XML documents, but also about what and how to evaluate!
- System architecture: DB on top of IR, IR on top of DB, true merging?
- Experimental evaluation of scoring functions and ranking algorithms for XML (INEX).
- Search over XML views: how to reuse existing indices?
- Search over a mix of HTML and XML.
- Joint scoring on full-text and scalar predicates.
- Score-aware algebra for XML for the joint optimization of queries on both structure and text.
- What do users want or expect? How do they interact?

DB and IR researchers should exchange expertise now!

Contact details

Sihem Amer-Yahia
AT&T Shannon Labs
NJ, 07932-0971, USA
http://www.research.att.com/~sihem
sihem@research.att.com
http://www.galaxquery.org/galatex

Mounia Lalmas
Department of Computer Science
Queen Mary University of London
http://www.dcs.qmul.ac.uk/~mounia
mounia@dcs.qmul.ac.uk
http://inex.informatik.uni-duisburg.de/