Database and Information Retrieval Techniques for XML

Ricardo Baeza-Yates
www.baeza.cl
Yahoo! Research &
UPF, Barcelona &
DCC, U. Chile

Mariano P. Consens
www.cs.toronto.edu/~consens
Information Engineering, MIE & Dept. of Computer Science
University of Toronto

Revised version of tutorials presented at
VLDB 2004, SIGIR 2995, ASIAN 2005

Agenda
1. Introduction
2. An Introduction to IR
3. Requirements for DB-IR
4. Semi-structured Data
5. Industrial DB-IR Examples: Oracle, Verity
6. DB Approaches
7. IR & Hybrid Approaches
8. Open Problems
9. Bibliography

1. Introduction
- Types of Data
- DB & IR Views
- Possible Solutions
- Applications
- Search Problems
- Tokenization

The (VLDB-only) DB-IR Saga

The (SIGIR Only) DB-IR Saga
The DB-IR Saga

Different Views on Data

Data and Databases

RDB vs. IR

The Notion of Relevance

Challenges in Current IR Systems

- **Data retrieval**: semantics tied to syntax
- **Information retrieval**: ambiguous semantics
- **Relevance**:
  - Depends on the user
  - Depends on the context (task, time, etc)
  - Corollary: The Perfect IR System does not exist
Possible Architectures

- IR on top of RDBs
- IR supported via functions in an RDB
- IR on top of a relational storage engine
- Middleware layer on top of RDB & IR systems
- RDB functionality on top of an IR system
- Integration via an XML database & query language

Problems of the IR view

- Very simple query language
  - Is it natural language the solution?
- No query optimization
- Does not handle the complete answer

Problems of the DB view

- The syndrome of the formal model
  - Model is possible because of structure
- The syndrome of “search then rank”
  - Large answers
  - Optimization is useless
  - Quality vs. Speed
  - E.g. XQuery
- What is a Database?
- Are RDBs really a special case of IR systems?
  - Full text over fields

Applications for Integrated Systems

- E-commerce search
- Intranets & enterprise data
- Customer support (e.g. CRM)
- News archives, bulletin boards, etc.
- Personal information (e.g. My Life Bits)
- P2P Web Search

Challenges posed by the Web

- Integration of autonomous data sources
  - Data/information integration
- Supporting heterogeneous data
  - How to do effective querying with both structured and text data
  - How to support IR-style querying on DBs
    - Because now users seem to know IR/keyword style querying more, even though structure is (good because it supports structured querying!)
    - How to support imprecise queries

Enterprise Search is Different

- Sophisticated systems run by librarians are morphing into simple self-service web-based search
  - Must be scalable, reliable, highly available
- Data is different
  - Heterogeneous in format & structure (documents, DBs, etc)
  - Less volume & better quality
- Searching is also different
  - Less & better queries, different tasks
  - Focus in recall rather than precision
    - Crucial in health, legal & police applications
- Other issues: security, able to search but not to see
What is a Bad Interface/Result?

- No search box
- Inability to judge user intent
  - No spell checking
  - No context disambiguation (cricket: game or bug?)
  - No recommendation system, no user feedback
- Too many hits: answer overload
  - Return 10,000 hits when most users looks only at the top-20
- The most relevant item is not at the top of the list
  - Act of faith
- Too many similar documents
  - Poor duplicate detection, poor clustering/category/naming
- Inability to understand why a document has been returned
  - No KWIC
- Lack of meta information
  - Size, format, date, etc.

Cost of a Bad Search

- Information is useless if no one can find it
  - Loss of employee productivity
  - Loss of customer satisfaction
  - Cost of people using out-of-date information
  - Cost of people using wrong information
  - Cost of recreating information which cannot be found
  - Cost of opportunity for not finding the information
  - Cost of generating wrong information

Bag-of-Words Representation

- Full-text continuum: ambiguity vs. completeness trade-off

Tokenization for structured data

- Mapping of values into strings, then into a sequence of substrings (or tokens)
- Tokenization is common practice when indexing text
- Can be applied to hierarchical naming schemes (URL), numbers, dates and times, codes (postal code, IP address, UPC)

Tokenization and XML Schema

XML Schema Datatypes supports token-friendly regular expression defined encodings

```
<complexType name="IP" base="string">
  <pattern value="([0-9]{1,3})\.(\d{1,3})\.(\d{1,3})\.(\d{1,3})" />
</complexType>
```

Agenda

1. Introduction
2. An Introduction to IR
3. Requirements for DB-IR
4. Semi-structured Data
5. Industrial DB-IR Examples: Oracle, Verity
6. DB Approaches
7. IR & Hybrid Approaches
8. Open Problems
9. Bibliography
Agenda
1. Introduction
2. An Introduction to IR
3. Requirements for DB-IR
4. Semi-structured Data
5. Industrial DB-IR Examples: Oracle, Verity
6. DB Approaches
7. IR & Hybrid Approaches
8. Open Problems
9. Bibliography

Challenges in Current IR Systems

Web Retrieval Architecture

Algorithmic Challenges

Text Similarity Models

Index

Vector model:
- words are dimensions
- $tf-idf$ is used for weights
- stopwords vs. rare words

Set Models:
- Boolean, Fuzzy sets, ...

Algebraic Models:
- Vector, LSI, etc.

Probabilistic Models:
- Probabilistic, inference & belief networks
Parallel Case

- Collection is divided per server
- Local indexes are used
  - Document partitioning
- Brokers distribute queries and merge results
- Simpler to build and update
- Good load balance, low concurrency
- Higher throughput by replication

- In theory a global partitioned index achieves higher concurrency but has lower load balance and more difficult to build & maintain

Non-word based Applications

- Suffix trees
- Linear building time
- Linear space (but larger than data)
- Suffix arrays
- Linear building time, less space
- Powerful search:
  - any substring
  - approximate search
  - regular expressions
- Applications: biology, music, linguistic, etc.

Link Ranking

- Incoming links count & variations (Li /Marchiori / Carriere et al. 1997; Joo & Myaeng, 1998)
- HITS (Kleinberg, 1998)
  - Authorities: good pages
  - Hubs: good links
- PageRank (Page & Brin, 1998)
  - Random walk + random jumps if “bored”
- Many variations of these ideas
- Good to find communities, spam, etc.
- Application to other problems (e.g. ranking relations)

Background Reading


Agenda

1. Introduction
2. An Introduction to IR
3. Requirements for DB-IR
4. Semi-structured Data
5. Industrial DB-IR Examples: Oracle, Verity
6. DB Approaches
7. IR & Hybrid Approaches
8. Open Problems
9. Bibliography

3. Requirements for DB-IR

- Motivating Applications
- Data and Query Requirements
- Sample Use Cases
Sample Paper on the Web

Bibliography Entry

<proceedings>
  <inproceedings>
    <author>Ricardo Baeza-Yates</author>
    <author>Gonzalo Navarro</author>
    <title>XQL and Proximal Nodes</title>
    ... 
  </inproceedings>
</proceedings>

- Describes metadata for the workshop article
- The XML data conforms to the DBLP schema (DTD)

Paper Content in XML

A Digital Library Application

- Web interface for the citation

Citations

Similar Documents

Applications Areas

- Scientific, Technical and Medical Reference Books, Journals, Publications
- Case Law and Litigation Materials
- Regulatory and Business Filings
- Maintenance, Repairs and Operations Manuals
- Product Documentation
  - Design
  - Procurement (SRM)
  - Customer Service (CRM)
- Collaboration, Portals
- Web, Intranet, Group & Personal Repositories
- Represents “80% of enterprise data”

Data Requirements

- Text, Documents, Images, Application Files
- Structured Data
  - Relations
  - Hierarchies
- Semi-structured Data
  - Editorial comments on the paper
- Assumption: XML provides a reasonably way to capture the requirements above
Publishing Relational Data

<table>
<thead>
<tr>
<th>USERS</th>
<th>USERID</th>
<th>NAME</th>
<th>RATING</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITEM</td>
<td>ITEMNO</td>
<td>DESCRIPTION</td>
<td>OFFERED BY</td>
</tr>
<tr>
<td>BID</td>
<td>USERID</td>
<td>BID_AMOUNT</td>
<td>BID_DATE</td>
</tr>
</tbody>
</table>

Queries on Views - Integration

<table>
<thead>
<tr>
<th>USERS</th>
<th>USERID</th>
<th>NAME</th>
<th>RATING</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITEM</td>
<td>ITEMNO</td>
<td>DESCRIPTION</td>
<td>OFFERED BY</td>
</tr>
<tr>
<td>BID</td>
<td>USERID</td>
<td>BID_AMOUNT</td>
<td>BID_DATE</td>
</tr>
</tbody>
</table>

Heterogeneous Sources - P2P

<table>
<thead>
<tr>
<th>&lt;users&gt;</th>
<th>&lt;user_tuple&gt;</th>
<th>&lt;userid&gt; 1243</th>
<th>&lt;name&gt; humphrey</th>
<th>&lt;rating&gt; ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;item&gt;</td>
<td>&lt;itemno&gt; 1066</td>
<td>&lt;description&gt; unicycle</td>
<td>&lt;offered_by&gt; ...</td>
<td></td>
</tr>
<tr>
<td>&lt;bid&gt;</td>
<td>&lt;userid&gt; 1243</td>
<td>&lt;itemno&gt; 1066</td>
<td>&lt;bid_amount&gt; ...</td>
<td></td>
</tr>
</tbody>
</table>

Query Requirements Overview

- Developing the web application
- Content-only
- Structure-only
- Content and Structure
- Relevance, Similarity
- Top-k

Proteomics Portal (courtesy T. Topaloglou, Protana)

- Relate proteins seen in experiments to the scientific literature

DB-IR Query Requirements

- Express arbitrary Full-Text (FT) searches
- Select the substructures where the FT condition applies (search context)
- Select the substructures to be returned (return context)
- Choose how to determine relevance for results and (weighted) queries
- Access and combine the relevance scores
- Limit answer to top-k
- Support approximate structural searches
  S. Amer-Yahia, N. Koudas, D. Srivastava, ICDE 2003 Tutorial
- Full composition of FT and structural queries
Additional DB-IR Requirements

- Efficient and scalable query evaluation, supported by
  - Indexes (FT and structural)
  - Optimizer (plans and operators)
- Rich functionality for presenting answers
  - Visual interfaces
  - Highlight the FT terms in context
- Support queries on integrated views
- Query heterogeneous structure
  - Within a single collection
  - In a data repository crawled from web sources
  - Across peer sources

Sample Use Cases

- Quick overview of the range of possible DB-IR requirements
  - Identify search and return contexts
  - Motivate relevance
  - Illustrate composition
- Extension of use cases from Full-text XQuery
  (//www.w3.org/TR/xmlquery-full-text-use-cases)

Finding Text in Elements

- Find all book titles containing the word "usability"
- Find all books with the phrase "usability tests" in book or chapter titles
  - Multiple search contexts, different return
- Find all books with the phrase "usability tests" (even across elements)
- Find all book titles for books with abstracts mentioning software developers (interpreted as having broad terms "software" near "developer")
  - Proximity
  - Thesaurus (developer, programmer)

Finding Text in Structure

- Find the first two sections mentioning "task" in chapters on "conducting usability tests" with the book abstract not mentioning "software"
  - Structured search contexts
    - book/chapter//section
    - book/chapter
    - book/abstract
  - Do the above ignoring footnotes in chapters but not in abstracts
  - Modifies the search contexts
  - Match the contexts approximately

Ranking

- Find how relevant to "usability" are the books
- Find the best two books on "usability tests"
  - Take into account reviewers comments
- Return all books with only the sections highly relevant to "usability"
- Rank on both approximate structure and content matching the sections mentioning "task" in chapters on "conducting usability tests" with the book abstract not mentioning "software"

Composing Queries

- For books with "usability" in the title create a flat list of all titles and the authors
- Find the 10 most relevant books about conducting "usability tests" which have more than one author and are published after "2000"
- Find all books published after "2001" which share a subject with the 10 most relevant books on "usability" that have titles mentioning "software" and "developer"
Agenda

1. Motivation
2. An Introduction to IR
3. Requirements for DB-IR
4. Semi-structured Data
5. Industrial DB-IR Examples: Oracle, Verity
6. DB Approaches
7. IR & Hybrid Approaches
8. Open Problems
9. Bibliography

4. Semistructured Data

- XQuery
- XQuery Evaluation
- Structured Text Models

XQuery History

XML Query Data Model

- Joint with XPath 2.0, XSL 2.0
  - Last version of Feb 2004
- Ordered, labelled forest
- Based on XML Information Set, PSVI
- Has node identity
- DTDs (from SGML, IR style)
- XML Schema (DB style)
  - Provide data types

XML Query Formal Semantics

- XQuery is a functional language
  - A query is an expression
  - Expressions can be nested with full generality
  - A pure functional language with impure syntax
- Static Semantics
  - Type inference rules
  - Structural subsumption
- Dynamic Semantics
  - Value inference rules
  - Define the meaning of XQuery expressions in terms of the XML Query Data Model

XQuery Expressions

- Element constructors
- Path expressions
- Restructuring
  - FLWOR expressions
  - Conditional expressions
  - Quantified expressions
- Operators and functions
- List constructors
- Expressions that test or modify data types
Path Expressions

XQuery uses the abbreviated syntax of XPath for path expressions --

document("bib.xml")

//author[last="Stevens" and first="W."]

document("bib.xml")//author

Full-text of Path Expressions

FLWOR Expressions

• FOR - LET - WHERE - ORDER BY - RETURN
• Similar to SQL’s SELECT - FROM - WHERE

for $book in document("bib.xml")//book
where $book/publisher = "Addison-Wesley"
return

<book>

{ $book/title,
   $book/author }
</book>

SQL vs. XQuery

"Find item numbers of books"

• SQL:

SELECT itemno
FROM items AS i
WHERE description LIKE ‘Book’
ORDER BY itemno;

• XQuery:

FOR $i IN //item_tuple
WHERE contains($i/description, "Books")
RETURN $i/itemno ORDERBY(.)

XQuery Implementations

• Software AG’s Tamino XML Query
• Microsoft, Oracle,
• Lucent Galax
• GMD-IPSI
• X-Hive
• XML Global
• SourceForge XQuench, Saxon, eXist, XQuery Lite
• Fatdog
• Qexo (GNU Kawu) - compiles to Java bytecode
• Openlink, CL-XML (Common Lisp), Kweetl,...
• Soda3, DB4XML and about 15 more

Inner Join

"List names of users and descriptions of the items they offer"

• SQL:

SELECT u.name, i.description
FROM users AS u, items AS i
WHERE u.userid = i.offered_by
ORDER BY name, description;

• XQuery:

FOR $u IN //user_tuple, $i IN //item_tuple
WHERE $u/userid = $i/offered_by
RETURN

<offering> {
   $u/name,
   $i/description }
</offering> ORDERBY(name, description)
Agenda
1. Introduction
2. An Introduction to IR
3. Requirements for DB-IR
4. Semi-structured Data
5. Industrial DB-IR Examples: Oracle, Verity
6. DB Approaches
7. IR & Hybrid Approaches
8. Open Problems
9. Bibliography

XQuery Research Implementations
- Native XML DBMS
  (e.g. Timber, Niagara, BEA/XQRL, Natix, ToX)
- XQuery systems
  (e.g. Galax, IPSI-SQ, XSM, MS-XQuery)
- XPath processors
  (e.g. XSQ, SPEX, XPush, Xalan, PathStack)
- Publish/subscribe
  (e.g. Y-Filter, IndexFilter, WebFilter, NiagaraCQ)
- Twig query processors
  (e.g. TwigStack, PRIX, TurboXPath)

Example XQuery and Pattern Tree
for $x$ in document("catalog.xml")//item,
$y$ in document("parts.xml")//part,
$z$ in document("supplier.xml")//supplier
where $x/part_no = $y/part_no
and $z/supplier_no = $x/supplier_no
and $z/city = "Toronto"
and $z/province = "Ontario"
return
$result
<$x/part_no>
<$y/description>
</result>

Example XQuery and Pattern Tree
or Twig Query

Stack Algorithms
- Region algebra encoding:
  - $T[DocID, Term, StartPos, EndPos, LevelNum]$ - elements
  - $[DocID, Term, TextValue, StartPos, LevelNum]$ - string values
- Stack algorithms: PathStack, TwigStack [BSK02]

Example XQuery Processing
for $x$ in document("catalog.xml")//item,
$y$ in document("parts.xml")//part,
$z$ in document("supplier.xml")//supplier
where $x/part_no = $y/part_no
and $z/supplier_no = $x/supplier_no
and $z/city = "Toronto"
and $z/province = "Ontario"
return
$result
<$x/part_no>
<$y/description>
</result>

Example XQuery Processing

XQuery Optimization in ToXop
- Path Summary Pruning
- Access Order Selection
- Path Summaries as Catalogs
ToXin Path Summary

For each distinct path in document there is a path in ToXin - is an exact path summary – reflects the structure of the document [RM01]

Initially proposed as a back-end - can answer any pattern queries

Incoming Path Summaries

Incoming-Outgoing Summaries

Outgoing Summaries

2-incoming Summaries

D(k) Summaries
Encodings, Summaries and Indexes

Agenda
1. Motivation
2. An Introduction to IR
3. Requirements for DB-IR
4. Semi-structured Data
5. Industrial DB-IR Examples: Oracle, Verity
6. DB Approaches
7. IR & Hybrid Approaches
8. Open Problems
9. Bibliography

6. DB Approaches
- IR on Relational Data
  - Keyword search
- IR on XML
  - Keyword search
  - Full QL + IR extension
  - Algebras and Evaluation

6-1. IR on Relational: Keywords
- BANKS
  - Gaurav Bhalotia, Arvind Hulgeri, Charuta Nakhe, Soumen Chakrabarti, S. Sudarshan, Keyword Searching and Browsing in Databases using BANKS, ICDE 2002
- DBXplorer
  - Sanjay Agrawal, Surajit Chaudhuri, Gautam Das, DBXplorer: A System for Keyword-Based Search over Relational Databases, ICDE 2002
- DISCOVER
  - Vagelis Hristidis, Yannis Papakonstantinou: DISCOVER, Keyword Search in Relational Databases, VLDB 2002

Keyword Search
- Keywords could be:
  - In the same tuple
  - In the same relation
  - In the Data or the Metadata
  - Connected through primary-foreign key relationships
- Results can be scored based on:
  - Distance of keywords within a tuple
  - Distance between keywords in # edges
  - IR-style ranking
  - Random walk probability (PageRank style)
  - Some combination of the above

Example Query [V. Hristidis]

<table>
<thead>
<tr>
<th>ORDER</th>
<th>CUSTOMER</th>
<th>NATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>10001</td>
<td>12312</td>
<td>USA</td>
</tr>
<tr>
<td>10002</td>
<td>12312</td>
<td>USA</td>
</tr>
<tr>
<td>10003</td>
<td>12312</td>
<td>USA</td>
</tr>
</tbody>
</table>

Results:

<table>
<thead>
<tr>
<th>Size</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>SHOP_{1} \to SHOP_{2}</td>
</tr>
<tr>
<td>4</td>
<td>SHOP_{1} \to SHOP_{2} \to SHOP_{3}</td>
</tr>
</tbody>
</table>

Smaller sizes usually denote tighter association between keywords.
6-2. IR on XML: Keywords

- **XKeyword**

- **XSearch**

- **XRANK**

**XSearch Example**

```
<proceedings>
  <inproceedings>
    <author>Moshe Y. Vardi</author>
    <title>Querying Logical Databases</title>
  </inproceedings>
  <inproceedings>
    <author>Victor Vianu</author>
    <title>A Web Odyssey: From Codd to XML</title>
  </inproceedings>
</proceedings>
```

The Content-Only Approach

Find papers by Vianu on the topic of "logical databases"

**Search:** Vianu logical databases

- Each document in the corpus is treated as a unit.
- A document containing some of the three query terms is considered as a result

**XQuery+FT Query Language**

```
FOR $i$ IN document("bib.xml")//inproceedings
WHERE $i$/author contains 'Vianu'
AND $i$/title contains 'Logical'
AND $i$/title contains 'Databases'
RETURN <result>
  <author> $i$/author </author>
  <title> $i$/title </title>
</result>
```

Requirements from the Search Tool

- A simple syntax that can be used by naive users
- Search results should include XML fragments and not necessarily full documents
- The XML fragments in an answer, should be semantically related.
  - For example, a paper and an author should be in an answer only if the paper was written by this author
- Search results should be ranked
- Search results should be returned in "reasonable" time

The document contains the three query terms. Hence, it is returned by a standard search engine. BUT

```
<proceedings>
  <inproceedings>
    <author>Moshe Y. Vardi</author>
    <title>Querying Logical Databases</title>
  </inproceedings>
  <inproceedings>
    <author>Victor Vianu</author>
    <title>A Web Odyssey: From Codd to XML</title>
  </inproceedings>
</proceedings>
```
XSEarch Query Syntax

• A query is a list of query terms
• A query term can be a
  – Keyword, e.g., database
  – Tag, e.g., inproceedings:
  – Tag-keyword combination, e.g., author:Vianu
• Optionally preceded by a ‘+’

The Example Revisited

• Find papers by Vianu on the topic of "logical databases"

logical +database inproceedings: author:Vianu

The keyword database must appear in the fragment of Vianu under the tag inproceedings, and the rank increases the rank of this fragment.

Example Revisited

XSEarch: author:Vianu title:

Good Result!
title and author elements ARE semantically related

Example (1)

Circled nodes belong to different inproceedings entities

Example (2)

Circled nodes belong to the same inproceedings entity
(see MLCAS in Li, Yu, Jagadish, Schema-Free XQuery, VLDB 2004)
Query Processing and Ranking

- Document fragments are extracted using indexes
- Extracted fragments are returned ranked by the estimated relevance

Result Ranking

Several factors increase the rank of a result
- Similarity between query and result
- Weight of labels appearing in the result
- Characteristics of result tree

TF-ILF
- Extension of TF-IDF, classical in IR
- Term Frequency: number of occurrences of a query term in a fragment
- Inverse Leaf Frequency: number of leaves containing a query term divided by number of leaves in the corpus

TF-ILF
- Term frequency of keyword $k$ in a leaf node $n_i$
  \[ tf(k, n_i) := \frac{\text{occ}(k, n_i)}{\max \{\text{occ}(k', n_i) \mid k' \in \text{words}(n_i)\}} \]
- Inverse leaf frequency
  \[ ilf(k) := \log \left(1 + \frac{1}{[n' \in N \mid k \in \text{words}(n')]}\right) \]

TF-ILF is the product between $tf$ and $ilf$

6-2. IR on XML: XQuery Text Search

- Conditions on Text
  - Equality: $//section[title="Procedure"]$
  - Full-text: $//section[contains(title, "Procedure")]]$

Full-text Requirements - I
- Full-text predicates and SCORE functions are independent
- Full-text predicates use a language subset of SCORE functions
- Allow the user to return and sort-by SCORE (0..1)
- SCORE must not require explicit global corpus statistics
- SCORE algorithm should be provided and can be disabled
- Problems:
  - Not clear how to rank without global measures
  - Many/no answers problems
  - Search then rank is not practical
  - How to integrate other SCORE functions?

Full-text Requirements - II
- Minimal operations:
  - Single-word and phrase search with stopwords
  - Suffix, prefix, infix
  - Proximity searching (with order)
  - Boolean operations
  - Word normalization, diacritics
  - Ranking relevance (SCORE)
- Search over everything, including attributes
- Proximity across markup elements
- Extensible
6-2. IR on XML: TeXQuery

TeXQuery Expressions

- Contains
  \[ \text{FTContentsExpr} := \text{ContextExpr} \text{ "ftcontains" } \text{FTSelection} \]
  returns true if a node in \text{ContextExpr} satisfies \text{FTSelection}
  
  \[ //\text{book} \text{"section ftcontains" ("usability" && "software") } //\text{title} \]

- Score
  \[ \text{FTScoreExpr} := \text{ContextExpr} \text{ "ftscore" } \text{FTWeightedSelection} \]
  returns a sequence of scores (for ranking and top-k)
  
  \[ //\text{book ftscore ("usability" weight 0.8 && $i/topic weight 0.2) } \]

TeXQuery Full-Text Model

QL-IR Design Choices

- SQL/MM structured text proposal
  - Functions have IR sublanguage as an argument, so the expression string can be constructed as a query
  - Explicit mark_subtexts() function supports highlighting matches

- TeXQuery
  - IR sublanguage grammar exposed and fully composable with XQuery
  - Implementation defined positions and scores

XQueryFT Query Plans

```xml
for $x$ in document("supplier.xml")//supplier,
  $y$ in document("catalog.xml")//item
  where $x/supplier_no = $y/supplier_no
  and $x/city = "Toronto"
  and $x/reviews ftcontains
    ("reliable" with stemming) && "on-time"
return
  <result>
    $y/name, $y/description
  </result>
```

More complex plans
6-2. IR on XML: TIX
Algebra

- TIX is an extension of the bulk XML algebra TAX that manipulates collections of scored trees with matching defined via scored pattern trees
  - S. Al-Khalifa, C. Yu, H. Jagadish, Querying structure text in an XML database, SIGMOD 2003

- Find document components in articles that
  - Are part of an article written by an author with last name “Doe” and are about “search engine”
  - Relevance to “internet” and “information retrieval” is desirable (but not necessary)

Find relevant document components in articles as before
For articles containing such components, find the reviews with similar titles

**Scoring:**

$\text{score} = \text{ScoringFunction}(['search engine'], ['internet', 'information retrieval'])$

**Threshold**

- Projection that retains input trees where at least one node has a top-k score, or a score higher than a threshold

**Pick**

- Projection that uses a condition with functions that can traverse the tree to remove redundant answers

**Operations implemented using stack-based algorithms on regions**
Query Evaluation with Relevance

R. Fagin, A. Lotem, M. Naor, Optimal aggregation algorithms for middleware. JCSS 2003 (Garlic System 1995)

• Threshold Algorithm
  – Given m sorted lists with object rankings
  – Aggregate the rankings from each list for each object
  – Return the top k ranked objects
  – Instance Optimal Solution: do sorted access (and the corresponding random access) until you know you have seen the top k answers

• IR Application: objects are document (fragments) and each list has the relevance of each document for a given keyword

Agenda

1. Motivation
2. An Introduction to IR
3. Requirements for DB-IR
4. Semi-structured Data
5. Industrial DB-IR Examples: Oracle, Verity
6. DB Approaches
7. IR & Hybrid Approaches
8. Open Problems
9. Bibliography

7. Hybrid & IR Approaches

• Overview of Approaches
• Retrieval Models
• Indexing
• INEX
• Ranking XML

Overview of Approaches

• RBD + IR: Two different APIs
• RDB + IR Hybrid: QUIQ, MOA, HySpirit, ...
• RBD “text search” accelerator
  – Text content is transformed to flat XML
  – XML is searched using an IR API
  – Results can be later combined with SQL
• IR System with SQL support
  – Special indexes for atomic data types
• XML Databases
  – Atomic data types as attributes (metadata)
  – Implementation on top of structured text models?

QUIQ (Kabra et al, 2003)

• Tuple: <tag-name, tag-type, tag-value>
• Query: match-filter-quality
  – Result: AND of match & filter
  – Match are approximate constraints
  – Filter are exact constraints
  – Relevance is adjusted by quality
• Indexing: built on top of a RDBMS
  – Non-text data is mapped to pseudo-words
  – Unified index & common TF-IDF model
  – Deferred update operations
• Evaluation: 60% faster than a RDBMS text extension

Retrieval Models

• Relational Model: DB2XML, XML-QL, TSIMMIS, LOREL
• Object-oriented Model: SOX, StruQL, ...
• Extended Vector Model
• Weighted Boolean Model: XQL, ...
• Probabilistic Model: XIRQL, ELIXIR, JunuXML, ...
Indexing

- Flat File: add information, SQL accelerators, ...
- Semi-structured:
  - Field based: no overlapping, Hybrid model, ...
  - Segment based: Overlapped list, List of references, p-strings
  - Tree based: Proximal Nodes, XRS, ...
- Structured:
  - IR/DB, Path-based, Position-based, Multidimensional
- Indexes:
  - Structure + Value index (XML on top of RDBs):
    - Toxin, Dataguides, T-indexes, Index Fabric, etc.
  - Integrated Full-text and Structure index:
    - Proximal Nodes, Region Algebra, String Indexing, ...

INEX

- Initiative for the Evaluation of XML
- Three types of tasks:
  - Content only search
  - Content & Structure Search
  - Clustering
- Started in 2002
- Cooperative relevance assessment
- About 40 groups per year

Ranking XML

- Content only:
  - exploit hierarchical structure
  - exploit importance of tags
- Content & structure:
  - Query languages with uncertainty & vagueness
  - Data types with vague predicates
  - Strict & fuzzy structural conditions
  - Dynamic $tf \times idf$

Integrated IR (Bremer & Gertz)

- Extension to XQuery
- Based on XML fragments
- Schemas are extended DataGuides
  - Enumeration of all rooted label paths
  - Ancestor relationships from structural joins
  - RANKBY operator
    - based on local & dynamic $tf-idf$
  - New node enumeration encoding
  - Path & term-index
    - Other smaller indexes (in total less than 60%)
  - More than 10 times faster than other XQuery prototypes

Agenda

1. Motivation
2. An Introduction to IR
3. Requirements for DB-IR
4. Semi-structured Data
5. Industrial DB-IR Examples: Oracle, Verity
6. DB Approaches
7. IR & Hybrid Approaches
8. Open Problems
9. Bibliography

8. Open Problems

- Heterogenous data
- Ranking tuples & XML
- New retrieval models
- DB issues for documents
- Simple/succinct vs. complex/verbose QL
  - Define an XQuery core?
- Query optimization and algebras
- Efficient algorithms
- Indexing & searching
- Quality evaluation (Web, XML)
Thank You

1. Motivation
2. An Introduction to IR
3. Requirements for DB-IR
4. Semi-structured Data
5. Industrial DB-IR Examples: Oracle, Verity
6. DB Approaches
7. IR & Hybrid Approaches
8. Open Problems
9. Bibliography (from course website)