XML Full-Text Search and Scoring

Sihem Amer-Yahia
AT&T Labs Research - USA
Database Department

Talk at the University of Tsukuba
May 10th, 2005
AT&T Labs

• 5,500 scientists and engineers

• AT&T’s patent portfolio includes 1,580 granted patents

• Over 80% of our scientists & technologists hold a PhD or other advanced degree

• Currently involved with approximately 90 U.S. & international universities
Research Priorities for 2005

- Speech
- Access
- Security
- Information Management
  - Network Management
  - Information Systems
- Enabling Infrastructure
  - Network
  - Software
Internet and Network Systems Research

- Optical Systems
- Access and Metro Network
- Access Technology and Applications
- Network Analysis and Optimization
- Network Theory
- Network Measurement, Monitoring, and Management
- IP Managed Services
- Network Security
- Internet and Network Incubation
Information & Software Systems

- Statistics
- Information Mining
- Customer Information
- Information Visualization
- Data Management
- Communication Software – Artificial Intelligence
- Dependable, Distributed Computing and Communication
- Software Systems - Scale
- Information Mining & Software Systems Incubation
- Information Mining Services
Voice & IP Services

- Speech Algorithms & Engines
- Natural Language Understanding & Dialog
- Knowledge Discovery from Speech & Data
- Multimedia Services
- IP Services
- Speech Services
Database Research at AT&T

- Researchers
  - Sihem Amer-Yahia, Mary Fernandez, Rick Greer, Ted Johnson, Flip Korn, Yannis Kotidis, Misha Rabinovich, Divesh Srivastava, Yannis Velegrakis

- Research
  - Large Scale Storage Systems
  - XML Query Processing
  - Data Stream Processing
  - Network Monitoring
  - Information Hosting
  - Data Integration

- Systems
  - Daytona, Silkroute, Galax, ShreX, GalaTex, Gigascope, Spider, Bellman, RaDar.
Motivation

- XML is able to represent a mix of structured and unstructured (text) information.

- Examples of XML repositories are: IEEE INEX data collection, Shakespeare's plays, DBLP, the Library of Congress collection.

- Existing query languages for XML (XPath and XQuery) are very limited when querying text.

- Two main activities: W3C, INEX (TREC for XML).
Outline

- Extending XPath/XQuery to support full-text search
  - syntax, data model and semantics.
- Scoring on both content and structure
  - approximate matching on structure.
- Open Issues.
Extending XPath/XQuery

- Full-text Search in XML
  - definition and requirements.
- TeXQuery and XQuery Full-Text
  - syntax, data model and semantics.
- GalaTex
  - architecture and implementation.

Joint work with Chavdar Botev and Jayavel Shanmugasundaram (Cornell U.), Emiran Curtmola (UC San Diego)
FT Search Queries

<book id="1000">
  <author>Elina Rose</author>
  <content>
    <p>The usability of software measures how well the software provides support for quickly achieving specified goals.</p>
    <p>The users must be and feel well-served.</p>
  </content>
</book>

- return book paragraphs containing the keyword “software” and stemmed forms of “usability” and do not contain “Rose”
- return elements containing "usability" and ("software" or "goals") within 12 words
- return ranked results
FT Search Definition

- **Context expression**: XML nodes where the search occurs: *e.g., book chapters.*
  - Use XPath and XQuery to identify nodes in the search context

- **Return expression**: document fragments that are returned to users: *e.g., book title and authors.*
  - Use XPath and XQuery to build returned answers

- **Search expression**: FT search conditions: *e.g., Boolean, proximity, stemming.*
  - Need for new language primitives

- **Score expression**: a scoring function for threshold or top-K queries.
  - Need for scoring framework
FT Search Requirements

- Composable FT search conditions
  - keyword, proximity order, times, stemming, stop words, thesaurus, wildcards, ...

- Integrate FT search with XPath/XQuery.
  - Non-trivial since structured XML queries operate on XML nodes, while FT queries operate on keywords and their positions within XML nodes.

- No extension to XPath/XQuery data model.

- Enable answer scoring and top-K queries.
Prior Work

- **SQL/MM** extends SQL with primitives on text, images and spatial data.

  - *Keyword similarity*: Elixir, XXL, XIRQL.
  - *Proximity distance*: InQuery, SQL/MM.
  - *Relevance ranking*: XQueryIR, XIRQL, XXL, JuruXML.
  - *Dynamic context*: XRank, TIX, XSearch.

- All explore only a few FT search primitives at a time and none of them develops a fully compositional model for FT search.
Prior Work

- Current XML query languages provide very rudimentary text/IR support:
  - Composability and extensibility not supported.
  - Data Model not formalized.
  - Hard to optimize across structured and unstructured queries.

- Sophisticated scoring and ranking algorithms interleaved with query processing.
XQuery in a Nutshell

- Input/Output: sequence of items
  - atomic types, elements, attributes, processing instructions, comments, ...
- XPath core navigation language.
- Variable binding.
- Element construction.
XQuery FLWOR Expression

- Find books on usability sorted on price:

```xml
for $item in //books/book
let $pval := $item/metadata/price
where fn:contains($item//content,"usability")
order by $pval ascending
return <result>
  {$item/title}
  <price> {$pval} </price>
</result>
```

- Limited sub-string operations: `fn:start-with()`, `fn:end-with()`
- No scoring or ranking
Alternative XQuery Extensions

- One function per FT primitive:
  
  \[
  \text{distance(}
  \quad \text{contains ($n$, "usability"),}
  \quad \text{contains ($n$, "software") or contains ($n$, "analysis"), 10}
  \text{)}
  \]

  returned type not sufficient to compute keyword distance.

- Sublanguage:
  
  \[
  \text{contains($n$, "usability and (software or analysis)
  distance 10")}
  \]

  FT search specified in an uninterpreted string that is opaque to the rest of the XQuery language.
TeXQuery

[Amer-Yahia, Botev, Shanmugasundaram WWW’04]

- Fully composable FT search primitives called FTSelections.
- Composable with XPath/XQuery.
- Formal data model called FullMatch.
- Scoring and ranking.
- Basis for XQuery Full-Text.
XQuery/TeXQuery Composability

Nest TeXQuery Expressions in XQuery Expressions

Evaluate to a Sequence of items

Nest XQuery Expressions in TeXQuery Expressions

Evaluate to a FullMatch
XQuery Full-Text

- Full-Text Task Force (FTTF) started in Fall 2002 to extend XQuery with full-text search capabilities.

- Members:
  - AT&T, IBM, Microsoft, Oracle, the US Library of Congress.


- FTTF documents (public comments are welcome!):
  - http://www.w3.org/TR/xmlquery-full-text-use-cases/
  - http://www.w3.org/TR/xmlquery-full-text-requirements/
  - http://www.w3.org/TR/xquery-full-text/
XQuery Full-Text Syntax

- **FTContainsExpr** ::= **RangeExpr** ( "ftcontains" **FTSelection** **FTIgnoreOption**? )?

  returns true if at least one node in **ContextExpr** satisfies **FTSelection**.

- **FTScoreClause** ::= "score" "$" **VarName** "as" **Expr**

  provides access to fine-grained ranking (e.g., threshold and top-k.)
FT Search: FTSelections

- FTWord | FTAnd | FTOr | FTNot | FTMildNot | FTOrder | FTWindow
  | FTDistance | FTScope | FTTimes | FTSelection (FTMatchOptions)*

1. books//title [. ftcontains “usability” ]

2. books//abstract [. ftcontains (“usability” || “web-testing”) ]

3. books//content [. ftcontains (“usability” && “software”) ordered window at most 3 ]

4. books//abstract [. ftcontains (“usability” && “web”) same sentence]

5. books//title [. ftcontains “usability” 4 occurrences ]

FT Search: FTMatchoptions

- FTCaseOption | FTDiacriticsOption | FTStemOption | FTThesaurusOption | FTStopwordOption | FTLanguageOption | FTWildCardOption

- books//title [. ftcontains ("usability") case sensitive with thesaurus "synonyms"]

- books//content [. ftcontains ("usability" && "software") with stopwords window at most 3 ]

- books//title ftcontains ("Utilisation" language "French" with stemming && ".?site" with wildcards )

- books//abstract [. ftcontains "usability" || "web-testing" with special characters ]
XQuery Full-Text Semantics

Propositional formula over word positions in DNF
FullMatch (AllMatch) Data Model

- FullMatch has a hierarchical structure represented as XML.
- Semantics of FTSelections specified as transformation of input FullMatches to an output XML FullMatch.
- Semantics of FTSelections specified in XQuery itself!
- FT search and structural search represented in same framework:
  - Enables joint optimization
Sample Document and Query

<book id='1000'>
  <author>Elina Rose</author>
  <content>
    <p>The usability of software measures how well the software provides support for quickly achieving specified goals.</p>
    <p>The users must not only be well-served, but must feel well-served.</p>
  </content>
</book>

books//book ftcontains
("usability" with stemming && "Rose") window at most 10
Query Evaluation Plan

Context expression → Scoring → Filtered context expression

Transform

FullMatch

FTDistanceAtMost (at most 10 words)

Build

FullMatch

FTAnd

FTWord Token: “usability”

FTWord Token: “Rose”

getPositions() on the inverted lists
The usability of software measures how well the software provides support for quickly achieving specified goals. The users must not only be well-served, but must feel well-served.
“usability” with stemming, “Rose”
FTWord Semantics

declare function fts:FTSingleSearchToken(
    $evalCtx as element()*,
    $searchToken as schema-element(fts:TokenInfo),
    $matchOptions as xs:string,
    $queryPos as xs:string )  as schema-element (fts:FullMatch)
{
    validate
    {
        <fts:FullMatch>
        {
            for $position in fts:getPositions($evalCtx, $searchToken, "")
            return
                <fts:SimpleMatch>
                    <fts:StringInclude queryString="{$searchToken/@word}"
                        queryPos="{$queryPos}"/>
                        {
                            $position
                        }
                    </fts:StringInclude>
                    <fts:SimpleMatch>
                </fts:SimpleMatch>
        }
        </fts:fullMatches>
    }
};
"usability" with stemming & & "Rose"

function fts:FTAnd ($fullMatch1 as element(fullMatch, fts:FullMatch),
                 $fullMatch2 as element(fullMatch, fts:FullMatch))
  as element(fullMatch, fts:FullMatch)
{<fullMatch> {for $sm1 in $fullMatch1/simpleMatch
            for $sm2 in $fullMatch2/simpleMatch
            return <simpleMatch> {$sm1/* $sm2/*} <simpleMatch>
   </fullMatch>}

05/10/2005
“usability” with stemming & “Rose”
“usability” with stemming && “Rose” window at most 10
Scoring Requirements

- Ranking answers on their relevance to FT query is an inherent part of FT search.
- Score value should reflect relevance of answer to FT expression.
- Score value depends on scoring algorithm but must satisfy:
  - Score is of type xs:float in the range [0, 1].
  - For score values greater than 0, a higher score must imply a higher degree of relevance.
  - If the Boolean evaluation of the FT expression is false then score value must be 0.
Scoring Examples

- **Query returning scores:**
  
  for $b$ in /books/book
  score $s$ as $b/content ftcontains "web site" && "usability" and $b//chapter/title ftcontains "testing"
  return $s

- **Top-K query example:**
  
  for $result$ at $pos$ in
    for $p$ in //books/book/paragraph
    score $s$ as $p ftcontains "users" && "software" with distance at most 13 words
    order by $s$
    return $p$
  where $pos$ <= 10
  return $result$
New Scoring Desiderata

- FT expression true implies score > 0.
- score = 0 implies ftcontains false.
- FT expression false should not imply anything for score.
- Answers satisfying query approximately may be returned.
- Scoring answers needs to consider scoring FT, scalar and structural predicates.
GalaTex overview

- First complete conformant implementation of W3C XQuery Full-Text language.
- Web demo includes W3C XQuery Full-Text Use Cases: [http://www.galaxquery.com/galatex](http://www.galaxquery.com/galatex)
- Poster presentation at WWW’05.
- Built on top of the Galax XQuery engine.
- Soon (in a few weeks) available as open source non-commercial software.
GalaTex Architecture

- Preprocessing & Inverted Lists Generation
- Full-Text Primitives (FTWord, FTWindow, FTTimes etc.)
- Galax XQuery Engine
- Equivalent XQuery Query
- GalaTex Parser

XML input:
```
<xml>
  <doc>
    Text Text Text Text
  </doc>
</xml>
```

Output:
```
<doc>
  Text Text Text Text
</doc>
```

Dependency:
- Preprocessing & Inverted Lists Generation
  - getPosition()
  - containsPos()
  - wordDistance()
XQuery Full-Text query is:

```xml
( : Q2: 2.2.2 Find all book subjects containing the phrase "usability testing" :)
<results>
{
  $xmlfile/books/book/metadata/subjects/subject[
    . ftcontains "Usability testing"
  ]
}
</results>
```

Generated XQuery query is:

```xml
<results>
{ $xmlfile/books/book/metadata/subjects/subject[
  let $ec_1 := ( . ) return
  fts:FTContains( $ec_1,
    fts:FTWordsSelectionAny( $ec_1, "Usability testing", validate
    {<fts:FTMatchOptions/>}, "1")))
}
</results>
```

Dynamic Evaluation:

```xml
<results xmlns:fts="http://www.w3.org/xquery-fulltext">
  <subject >Usability testing</subject>
  <subject >Usability testing</subject>
  <subject >Usability testing</subject>
  <subject >Usability testing</subject>
</results>
```
Scoring XML

Joint work with Nick Koudas, (U. of Toronto), Amélie Marian (Columbia University), Divesh Srivastava (AT&T Labs Research), David Toman (University of Waterloo)
Motivation

- Queries on XML data combine conditions on structure with conditions on values.

- Computing the relevance of an answer to a query should rely on:
  - Evaluating conditions on values and on structure approximately.
  - Combining scores.

- Only a few recent contributions to approximate XML queries on structure [Schlieder’02, Delobel and Rousset’02, Amer-Yahia et al’02].

- **Goal**: Study query approximation on structure in XML and define a family of scoring methods.
Outline

- Examples of Query Approximation on Structure.
- XML Query Relaxation.
- Scoring Functions for XML.
Heterogeneous XML Data about books

Query:

book ./info/author ftcontains “Dickens”] and ./info/title ftcontains “Expectations”] and ./edition ftcontains “paperback”]
XML Query Relaxation

[Amer-Yahia, Cho and Srivastava EDBT’02]
[Amer-Yahia, Lakshmanan and Pandit SIGMOD’04]

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

Query

Data

book

info

edition (paperback)

author Charles Dickens

title Great Expectations

book

info

edition?

author C. Dickens

title Great Expectations

book

info

edition paperback

author Dickens

title Great Expectations
XML Query Relaxation

[Amer-Yahia, Cho, Srivastava EDBT'02]

- Encode relaxations in a single join plan:
  - More efficient than rewriting-based techniques.
  - Static threshold for top-K pruning.
  - Batch mode processing.
- Challenge: Maximize answer scores to enable early pruning
- Traditional join ordering not applicable
Scoring Functions Critical for Top-k Query Processing

- Top-k answer quality depends on scoring function.
- Efficient top-k query processing requires scoring function:
  - Monotonic.
  - Fast to compute.
- Little attention given to scoring functions for structured and semi-structured data
  - Extensively studied over text data (e.g., $tf.idf$)
  - Proposed scoring function inspired by $tf.idf$ for XML data
Adaptation of *tf.idf* to XML

[M Marian, Amer-Yahia, Koudas, Srivastava ICDE’05]

<table>
<thead>
<tr>
<th>Document Collection (Information Retrieval)</th>
<th>XML Document</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document</td>
<td>XML Node (result is a subtree rooted at a distinguished node, i.e., a node with a given label and structural properties)</td>
</tr>
<tr>
<td>Keyword(s)</td>
<td>Query Pattern</td>
</tr>
<tr>
<td><em>idf</em> (<em>inverse document frequency</em>) is a function of the fraction of documents that contain the keyword(s)</td>
<td><em>idf</em> is a function of the fraction of distinguished nodes that match the query pattern</td>
</tr>
<tr>
<td><em>tf</em> (<em>term frequency</em>) is a function of the number of occurrences of the keyword in the document</td>
<td><em>tf</em> is a function of the number of ways the query pattern matches the distinguished node</td>
</tr>
</tbody>
</table>
Scoring Function for XML Approximate Matches

- Required properties:
  - Exact matches should be scored higher than relaxed matches (\(idf\))
  - Returned elements with several matches should be ranked higher than those with fewer matches (\(tf\))

- How to combine \(tf\) and \(idf\)?
  - \(tf.idf\), as used by IR, violates above properties
  - Ranking based on \(idf\), then breaking ties using \(tf\) satisfies the properties

\[
\text{score}(a) \leq \text{score}(b)
\]
A Family of Scoring Methods for XML Path Queries

- **Twig predicate**
  - High quality
  - Expensive computation

- **Path predicates**

- **Binary predicates**
  - Low quality
  - Fast computation

---

Query

```
book
  info
    edition
      (paperback)
    author
      (Dickens)
  title
    (Great Expectations)
```

```
  info
    info
      edition
        (paperback)
      author
        (Dickens)
    title
      (Great Expectations)
```

```
  author
    title
      info
        edition
          (paperback)
        (Dickens)
      (Great Expectations)
```
Representing Relaxed Query Patterns: DAG Structure

- Each child is more relaxed (has more matches) than its parents.
- $idf$ of a child is lower than the $idf$ of its parents.
- $idf$ scores are accessible in constant time for any match (complete or partial) using hash function.

Exhaustive algorithm to build the DAG.
Query Processing using the DAG

- **Benefits:**
  - Score computation done in a **preprocessing** phase (using exact or approximate information)
  - Score access during query processing done in **constant time**
  - Additional information needed for query processing precomputed and accessed in constant time (e.g., score upper bound)
- **tf** estimated at runtime based on available information
Summary of Contributions

- A family of scoring methods for XML queries
  - Structure and content
  - Structural relaxations
- Evaluation of the scoring methods tradeoffs.
- Efficient data structures to compute and access scores during top-k query processing.
Open Issues

- Extensive experimental evaluation of scoring functions and ranking algorithms for XML:
  - INEX topics and datasets.
  - In collaboration with K. Hatano (NAIST).
- Define a score-aware algebra for XQuery Full-Text for the joint optimizations of queries on both structure and text:
  - Consistent scoring: equivalent query expressions should result in same scores.
  - Consistent ranking: equivalent query expressions should result in the same topK results for any given document fragment.
  - Optimize individual FTSelections: e.g. FTAnd.
  - In collaboration with E. Curtmola and A. Deutsch (UCSD).
- Refine XQuery Full-Text language syntax and semantics:
  - A syntax for specifying structural relaxations?
  - Semantics of structural relaxations when combined with query approximation on content?