TeXQuery: a Full Text Extension to XQuery

Rui Xi
University Konstanz
xi@inf.uni-konstanz.de
Motivation

• XML is able to represent both structured and unstructured (text) data.
• Two compelling paradigms for querying XML documents
  **Database-style query languages**: XPath provides powerful primitives to navigate document structure.
  **IR-style querying**: Keyword/full-text search provides powerful search primitives at the fine level of element and attribute content.
• Existing query languages such as XQuery/Xpath are limited when querying text. (e.g. phrase matching, order specification, stemming, etc)
Querying an XML in DB and in IR

XQuery/XPath engines

IR engines
### Motivation (contd)

<table>
<thead>
<tr>
<th>Querying Tools</th>
<th>Structure</th>
<th>Text</th>
<th>Scoring results</th>
</tr>
</thead>
<tbody>
<tr>
<td>IR engines (i.e. Google, etc.)</td>
<td>Simple path expressions</td>
<td>Power text search</td>
<td>Powerful scoring using well established model (TF*IDF)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Efficient indices and algorithms</td>
<td></td>
</tr>
<tr>
<td>XPath 2.0, XQuery 1.0</td>
<td>Powerful tree manipulation primitives</td>
<td>Limited substring matching (i.e. contains, start-with, end-with) + string manipulation functions</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Power “Return” Clauses</td>
<td>Coarse data model</td>
<td></td>
</tr>
<tr>
<td>XQuery Full-Text</td>
<td>Powerful tree manipulation primitives</td>
<td>Fine-grained model (at the level of words)</td>
<td>Scoring on both structure and text</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Powerful and fully composable FT search primitives</td>
<td>Extend TF-IDF for XML tree based structure</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Efficient query evaluation for both structure and contents</td>
<td></td>
</tr>
</tbody>
</table>
Full-Text 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>

**Example Queries:**

- "software" && "usability" with stems && ! "Rose"
- "usability" && ("software" || "goals") distance 12
Full-Text Search in XML

- **Context expression**
  defines nodes where search occurs: *e.g.* book chapters.
- **Return expression**
  defines document fragments that are returned to users: *e.g.*, book title and authors.
- **Search expression**
  defines FT search conditions: *e.g.*, boolean, proximity, stemming.
- **Score expression**
  defines an expression that might be used to score returned fragments.
XQuery in nutshell

• Functional language.
• Input/Output: sequence of items (atomic types, elements, attributes, processing instructions, comments, ...).
• Fully compositional.
• Variable binding.
• XPath core navigation language.
• Element construction (return clause).
XQuery FLWOR Expression

- Find the title and price of books on usability and sort books from the cheapest to the most expensive.
  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>
FT Search Design Goals

• Identify **basic FT search primitives** natural to querying XML.

• Primitives should be **composable** with each other to express arbitrarily complex FT conditions.

• Seamlessly **integrate regular XQuery with FT search** to query over both structured and full-text data. Non-trivial because structured XML queries operate on XML nodes, while FT queries operate on keyword search tokens and their positions within XML nodes.

• **Avoid** any extension to the XPath and XQuery data model.

• Define **ranked results** in order to support threshold and topK queries.
Alternative Solution: functions

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

- "contains" returns Boolean values. Not enough to compute distance.

- Extra information about search tokens and their positions needs to be “carried around” with the Boolean value.

- **Problem:** Fundamental extension to the XQuery data model violating design goals.
Alternative Solution: sublanguage

```
contains($n, "usability and (software or analysis) distance 10")
```

- Embed entire FT search in a single `contains` function.
- No extension to XQuery data model is needed.
- **Problem:** FT search specified in an uninterpreted string that is opaque to the rest of the XQuery language.
- **Solution:** Make string conform to a well-defined grammar and composable with itself and with XQuery.
Discussions

• The functional based syntax has the limitations in meeting design goals.

• It’s unusual because the precise syntax should not impact the expressive power.

• The problem is that we are proposing an extension to XQuery and syntax should fit within the framework.
TeXQuery in a nutshell

- Introduce two new XQuery expressions, called TeXQuery expressions that seamlessly integrates with XQuery

- Provides a set of powerful FT search primitives called FTSelections.

- FTSelections are **fully composable**.

- Relies on a formal data model called **FullMatch** that based on the positions of linguistic tokens *within* XML nodes.

- Permits scoring and ranking.
TeXQuery primitives

• Two TeXQuery expressions: FTContainsExpr and FTScoreExpr.

• \textbf{FTContainsExpr::= ContextExpr “ftcontains” FTSelection}
  returns true if at least one node in ContextExpr satisfies FTSelection.

• \textbf{FTScoreExpr::= ContextExpr “ftscore” FTWeightedSelection}
  returns a sequence of scores. Provides access to fine-grained ranking (e.g., threshold and top-k.)
FTContainsExpr: FTSelection

- FTContainsExpr ::= ContextExpr ”ftcontains” FTSelection

FTSelection ::= FTStringSelection |
               FTAndConnective |
               FTOrConnective |
               FTNegation |
               FTMildNegation |
              FTOrderSelection |
               FTScopeSelection |
              FTDistanceSelection |
               FTWindowSelection |
               FTTimesSelection |
               FTSelection (FTContextModifier)*
FTContainsExpr: FTContextModifier

- **FTSelection ::= FTSelection (FTContextModifier)***

- FTContextModifier defines the FTS environment, which can modify the operational semantics of FTSelection such as stemming, stopwords, diacritics and case.

- **FTContextModifier ::= FTCaseCtxMod | FTDiacriticsCtxMod | FTSpecialCharCtxMod | FTStemCtxMod | FTThesaurusCtxMod | FTStopWordCtxSpec | FTLanguageCtxMod | FTRegExCtxMod | FTIgnoreCtxMod**
FTScoreExpr:FTSelectionWeights

• FTScoreExpr only provides the framework of supporting different scoring mechanisms
• It also provides the necessary language interface for specifying the weights in query
  - //book ftscore ‘usability’ weight 0.8 && ‘testing’ weight 0.2
• Two high-level properties that every scoring measure should satisfy:
  - The score of a node should be zero iff the node not satisfy the FT condition.
  - A higher value of the score imply a higher relevance to FTSelection With Weights.
Integration with XQuery

- **Simple Example:**
  
  ```xquery
  for $book in 
    books/book ftcontains "usability" with stems && "software" && !"Rose"
  return <hit>
    $book
  </hit>
  ```

- **Top-K Example:**
  
  ```xquery
  for $hit at $i in 
    for $book in books//section ftcontains "usability"
    let $score := $book ftscore "software" weight 0.7
    order by $score descending
  return <hit>$book<score>$score</score></hit>
  where $i < 20
  return $hit
  ```
Rationale for TeXQuery

- XQuery is defined on XML structures.
- FTS operates on linguistic units (e.g., words, sentences) not represented in the XQuery 1.0 and XPath 2.0 Data Model.
- Basic FTS operators and the way they are composed is different from XQuery operators (e.g., 'and' and 'or' in XQuery are Boolean, but their full-text equivalents are operators on linguistic units).
- FTS uses token positions (e.g., textual proximity).
- TeXQuery introduces new operators for the full-text sublanguage and integrates them to XQuery.
TeXQuery Data Model

• XQuery expressions take sequence(s) of nodes as input and evaluate to a sequence of nodes.
• FTSelection takes FullMatch(es) as input, and evaluates to a FullMatch in the FTS data model.
• FullMatch captures linguistic token positions, and other information required for full composability of FTSelections.
XQuery and TeXQuery: Composability

- TeXQuery expressions define a well-formed mapping between the fullmatch data model and XQuery data model.
- Fullmatch is only internal to TeXQuery.
- TeXQuery expressions still return a sequence of items that are fully composable with XQuery data model.
Composability (contd)

- **Conversion from FullMatch to XQuery data model:**
  To enable TeXQuery expressions to be nested and composed with regular XQuery expressions, three new XQuery expressions for FTS which convert a FullMatch to a sequence of items are proposed.

- **Conversion from XQuery data model to a FullMatch:**
  We use the result of an XQuery expression as a search token in an FTSelection by converting the XQuery expression to the FullMatch associated with that search token.
Scores and Ranking: Issues

• Use a ranking expression that is different from the search expression. The ranking expression can be as complex as the search expression. E.g. we search for documents on “AIDS” and rank them by their relevance to “health benefits”.

• Provide ability to express user weights in scoring expression.

• Rank on both scalar and FT predicates.
Scores and Ranking in TeXQuery

- *ftscore* returns a sequence of floats.

- Scoring function is implementation-dependent but must satisfy:
  - The scoring function should produce score values in the range 0-1 (inclusive)
  - If context node does not satisfy FTSelection used in *ftscore*, score is 0
  - If context node satisfies FTSelection used in *ftscore*, score should be [0,1] (use this condition in where clause to filter out answers).
  - For context nodes that satisfy the FTSelection, a higher score implies a higher relevance to the FTSelection.
FullMatch Data Model

• XQuery is a “sequence of items” data model and its granularity on XML node is inadequate for the full composability of FTSelections.
• FullMatch data model based on the positions of linguistics tokens within XML nodes.
• A position include the following attributes and can be modeled as an XML element confirming to the following DTD.

<!ELEMENT Position(Token, Identifier, Node, Sentence, Para, Context)>
Positions Example

- `<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 feel well-served.</p>`
- `</content>`
- `</book>`

//book ftcontains "software" && "usability" with stems
FullMatch data model (contd)

- FM essentially a first-order logic DNF predicate
- Each Simplematch corresponds to one of the disjuncts in the DNF formula.
- The XML schema of FullMatch model
  ```xml
  <!ELEMENT FullMatch(SimpleMatch)*)>
  <!ELEMENT SimpleMatch(StringInclude|StringExclude)*)>
  <!ELEMENT StringInclude Position>
  <!ELEMENT StringExclude Position>
  ```
“Software” example

- `<book(1) id(2)="1000(3)">
  <author(4)>Elina(5) Rose(6)</author(7>)
  <content(8)>
  <p(9)> The(10) usability(11) of(12) software(13) measures(14) how(15) well(16) the(17) software(18) provides(19) support(20) for(21) quickly(22) achieving(23) specified(24) goals(25) </p(26)> ...
</content(8)>

AllMatch

Match

Match

StringInclude: "software"

Pos 13

Pos 18
"usability" with stems example

- <p>The usability of software measures how well the software provides support for quickly achieving specified goals. The users must be and feel well-served.</p>
"usability" with stems & & "software"

• <p> The usability of software measures how well the software provides support for quickly achieving specified goals. The users must be and feel well-served.</p>
Example query = find all book paragraphs that contain “software” and “users” at a distance at most 13 words of each other.
FT Predicates – example

- GalaTex data model: FullMatches
  - positions of tokens within XML nodes
  - all possible matches
  - XML representation
- Input query: all “users” in book contents

```xml
<book id="1000">
  <author>Mary Rose</author>
  <content>
    <p>The usability of software measure how well the software provides support for quickly achieving specified goals for users. </p>
    <p>The users must be and feel well-served.</p>
  </content>
</book>
```
FT Predicates – example

- FTAnd + FTDistance: “software” and “users” at distance at most 13

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

- FTAnd + FTDistance: “software” and “users” at distance at most 13

FTDistanceAtMost

FTAnd

FTWordsSelectionAny

FTWordsSelectionAny

getPositions in the inverted lists

AllMatch

StrInclude “software”
Pos 13

StrInclude “users”
Pos 28

StrInclude “software”
Pos 13

StrInclude “users”
Pos 32

StrInclude “software”
Pos 18

StrInclude “users”
Pos 28

StrInclude “software”
Pos 18

StrInclude “users”
Pos 32
Architecture

Preprocess the input text

getPositions() containsPos() wordDistance()

Full-Text semantic functions implementation in XQuery

eval.

GalaTex

[Philip Brown]

XQuery query

Full-Text Query

GalaTex Parser

inverted lists

.xml

.txt

Full-Text Query

.Equivalent XQuery query

.xml

.xq

.xq
Query Evaluation Tree

```
Query Evaluation Tree

True/False
  /\  
FTContains

FullMatch
  /\  
FTOrderSelection

FullMatch
  /\  
FTDistanceSelection

FullMatch
  /\  
FTAndConnective

FTStringSelection searchToken

FTStringSelection searchToken

FTStringSelection searchToken
```
Lessons

• A sublanguage approach is necessary to achieve full composability.
• Simple and powerful scoring mechanism.
• Clean Semantics and well-defined default behavior for easy extensibility.
• No extension to the XQuery data model should be necessary.
TeXQuery Highlights

- Sublanguage is composed of atomic FTSelections that are fully composable (any number of times!).
- Support for thesauri, ability to share modifiers, specify/override stopwords, control number of occurrences,
- Scoring expression allows user weights.
- Evaluation model is represented in an XML Schema.
- Semantics of TeXQuery can be specified in XQuery!
- Does not require any change to XQuery constructs and data model.
A Quick Summary of W3C Effort

• Full-Text Task Force (FTTF) started in Fall 2002 to extend XQuery with full-text search capabilities: IBM, Microsoft, Oracle, the US Library of Congress.
• FTTF documents published on February 14, 2004.
  http://www.w3.org/TR/xmlquery-full-text-use-cases/
  http://www.w3.org/TR/xmlquery-full-text-requirements/
• XQuery Full-Text highly influenced by TeXQuery.
• A working draft describing the syntax and semantics of the XQuery Full-Text on July 9, 2004 at:
  http://www.w3.org/TR/xquery-full-text/