A TeXQuery-Based XML Full-Text Search Engine

Chavdar Botev
Cornell University
cbotev@cs.cornell.edu

Sihem Amer-Yahia
AT&T Labs–Research
sihem@research.att.com

Jayavel Shanmugasundaram
Cornell University
jai@cs.cornell.edu

ABSTRACT

We demonstrate an XML full-text search engine that implements the TeXQuery language. TeXQuery is a powerful full-text search extension to XQuery that provides a rich set of *fully composable* full-text primitives, such as phrase matching, proximity distance, stemming and thesauri. TeXQuery enables users to seamlessly query over both structure data and text, by embedding full-text primitives in XQuery and vice versa. TeXQuery also supports a flexible scoring construct that scores query results based on full-text predicates and permits top-k queries. TeXQuery is the precursor of the full-text language extension to XPath 2.0 and XQuery 1.0 currently being developed by W3C.

1. INTRODUCTION

One of the key benefits of XML is its ability to represent a mix of structured and unstructured (text) data. This is illustrated in many existing XML data repositories such as the IEEE INEX data collection, Shakespeare's plays in XML, the Library of Congress documents in XML, and SIGMOD Record in XML. In addition, many applications such as library science have a growing need to support a mix of structured and full-text queries over these document collections.

While XQuery and its core navigation language, XPath, provide powerful structured queries over XML documents, they have only limited rudimentary capabilities for querying the unstructured data in XML documents using full-text search. These capabilites are primarily based on the *contains* function. The expressiveness of the *contains* function is limited to simple keyword and phrase matching and cannot express sophisticated text search primitives such as Boolean queries, proximity distance, order specification, stemming and thesauri. In addition, the *contains* function cannot score query results which is necessary to compute the relevance of query answers when querying textual content in documents.

On the other hand, as it can be seen from the W3C Full-Text Use Cases Document [2], there is a significant number of practical usages of queries that combine structured and unstructured search. As an illustration, consider the following use-case from [2]: *Find all 'book' XML elements that contain the keywords 'usability' and 'software' within three keywords of each other, and the keyword 'Rose'; further use stemming for the keyword 'usability' and case-sensitivity for the keyword 'Rose'; return the 'title' child sub-element and the 'author' descendant sub-elements of these 'book' elements*. Clearly, this query involves both structured search (books, titles and authors of books) and full-text search on the content of the books.

To address the inability to express the above query (and other complex full-text search queries) in XQuery, we have designed and implemented TeXQuery. TeXQuery is a full-text search extension to XQuery. It supports powerful set of fully composable full-text primitives, supports a flexible scoring construct that can return top-k results, and supports queries over both structured data and text.

Designing a set of fully composable full-text primitives that are tightly integrated with structured XQuery queries is a non-trivial task because structured XML queries operate on items (e.g., element nodes and attribute nodes), while by their very nature, full-text queries operate on tokens and their positions *within* XML nodes. TeXQuery addresses this issue by providing a set of full-text search primitives, called *FTSelection*s, that rely on a formal model, called *FullMatch*. *FullMatch* represents search tokens and their positions in an XML document. Each *FTSelection* takes zero or more *FullMatch*es and produces a *FullMatch*. Thus, *FTSelection*s can be arbitrarily composed, allowing complex full-text searches to be specified. The semantics of the *FullMatch*es can be described using XQuery functions as shown in [1]. This design fully integrates full-text search into XQuery without requiring to modify the XQuery data model and formal semantics [4].

Our model also allows for customizable relevance scoring by allowing weights to be associated with each *FTSelection*.

To the best of our knowledge, TeXQuery is the first language and implementation that provides such an integrated querying of structure and text in XML documents. TeXQuery is powerful enough to express every use case in the W3C Full-Text Use Cases Document [2], satisfies the Full-Text Requirements in [3]. TeXQuery is the precursor of the full-text language extension to XPath 2.0 and XQuery 1.0 currently being developed by W3C.

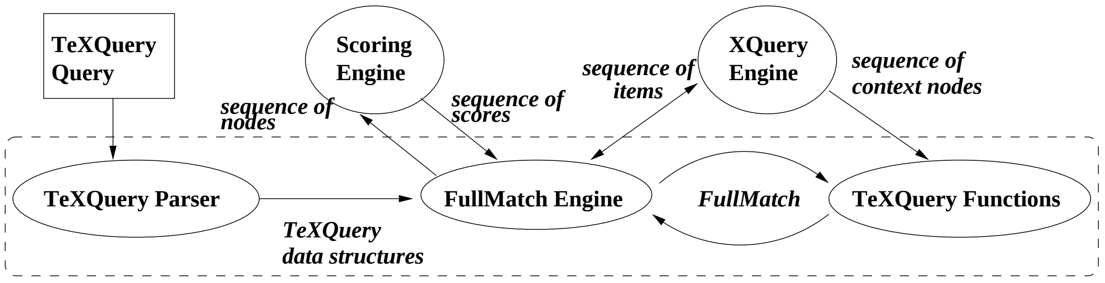

Figure 1: Architecture of the TeXQuery Engine

2. DEMONSTRATION OVERVIEW

We demonstrate a complete implementation of the TeXQuery in the context of the Quark system [1]. The Quark system is implemented in C++ and is capable of running both regular XQuery queries as well as a mix of TeXQuery and XQuery queries. Using this system, we will demonstrate the following features of TeXQuery:

Visualizing Input documents: Users can visualize and select input documents to be queried. We provide preloaded XML documents including Shakespeare plays, SIGMOD Record, the 500MB IEEE INEX Collection (which contains IEEE computer science publications), and the XML documents used in the W3C Full-Text Use Cases Document [2]. For the documents collections with DTDs, users can visualize the corresponding DTDs.

Aids to Query Specification: Users can specify queries in three ways. First, they can choose from a variety of sample preloaded TeXQuery queries, including all the queries in the W3C Full-Text Use Cases Document. Second, they can write their own TeXQuery query in a specified window. Finally, they can input their queries in a form interface in which they specify: (i) The **context expression** as an XPath/XQuery expression whose result is a set of element nodes that identify the context in which the full-text expression is applied; (ii) The **search expression** that specifies the full-text conditions combined with any XQuery condition; (iii) The **return expression** as in XPath/XQuery to identify expected answers; (iv) The **score expression** as a weighted full-text condition that will be used to assign scores to query answers; (v) The **value of K or threshold** if the user is interested in top-K answers or answers whose score exceeds a certain threshold. If the user does not specify a scoring expression, answers are returned in document order.

Visualizing the Evaluation Plan: While a query is being evaluated by the TeXQuery engine, users can visualize its evaluation plan. The system displays a graphical representation of *FullMatch*es at each step of the evaluation process which allows users to follow every step of the query evaluation.

Answer Explanation: An element or attribute node qualifies as a query answer if it satisfies the full-text condition specified in the query. The system displays all the hits found along with each node. A query result is converted into an HTML document in which hits are highlighted and answers are ranked by document order or by relevance order.

[1]http://www.cs.cornell.edu/database/Quark

3. SYSTEM ARCHITECTURE

Figure 1 depicts the architecture of the TeXQuery implementation. At the core of our implementation, is the TeXQuery engine that interacts with the Quark XQuery engine through a set of functions, one for each *FTSelection*. These functions take the XML representation of one or more *FullMatches* and return the XML representation of a *FullMatch*.

When a TeXQuery query is entered, the system parses the query and first identifies and evaluates any nested XQuery expressions in the full-text query. The TeXQuery query evaluation proceeds as follows. First, for each search term in the query, using inverted list indices, a *FullMatch* is generated that contains the positions of the full-text terms in the input document. These *FullMatch*es are then composed by the functions that implement each *FTSelection*. When the final *FullMatch* is built, it is used to filter the context nodes evaluated in XQuery and returns qualified context nodes as answers. If needed, relevance scoring is performed by the scoring engine that uses the weights specified in the query. In this case, the relevance scores are returned as a sequence of floats to the XQuery engine that can be used to rank the query answers.

4. REFERENCES

[1] S. Amer-Yahia, C. Botev, and J. Shanmugasundaram. TeXQuery: A Full-Text Search Extension to XQuery. WWW Conference, New York, New York, 2004.

[2] The World Wide Web Consortium. XQuery and XPath Full-Text Use Cases. W3C Working Draft. Available at http://www.w3.org/TR/xmlquery-full-text-use-cases/, Feb. 2003.

[3] The World Wide Web Consortium. XQuery and XPath Full-Text Requirements. W3C Working Draft. Available from http://www.w3.org/TR/xmlquery-full-text-requirements/, May 2003.

[4] The World Wide Web Consortium. XQuery 1.0 and XPath 2.0 Data Model. Working Draft. Available from http://www.w3.org/TR/xpath-datamodel/, May 2003.