INTERNATIONAL JOURNAL OF RESEARCH IN COMPUTER APPLICATIONS AND ROBOTICS
ISSN 2320-7345

EFFECTIVE KEYWORD SEARCH OF FUZZY TYPE IN XML

Mr. Mohammed Tariq Alam[1] ,Mrs.Shanila Mahreen [2]
Assistant Professor in Electronic & Communication Engineering at Nawab Shah College Of Engineering & Technology (Affiliated to JNTUH), Malekpet, Hyderabad-500024, A.P, India.

ABSTRACT:

In a traditional keyword-search system over XML data, a user composes a keyword query, submits it to the system, and retrieves relevant answers. In the case where the user has limited knowledge about the data, often the user feels "left in the dark" when issuing queries, and has to use a try-and-see approach for finding information. In this paper, we study fuzzy type-ahead search in XML data, a new information-access paradigm in which the system searches XML data on the fly as the user types in query keywords. It allows users to explore data as they type, even in the presence of minor errors of their keywords. Our proposed method has the following features: 1) Search as you type: It extends Auto complete by supporting queries with multiple keywords in XML data. 2) Fuzzy: It can find high-quality answers that have keywords matching query keywords approximately. 3) Efficient: Our effective index structures and searching algorithms can achieve a very high interactive speed. We study research challenges in this new search framework. We propose effective index structures and top-k algorithms to achieve a high interactive speed. We examine effective ranking functions and early termination techniques to progressively identify the top-k relevant answers. We have implemented our method on real data sets, and the experimental results show that our method achieves high search efficiency and result quality.

KEYWORDS: Fuzzy, Framework, XML.

INTRODUCTION

The application concentrates on performing searching in XML using fuzzy search over type ahead search. The system allows the user to identify a XML file in whose content search has to be made. The user has a facility to have the XML file created at runtime from SQL Server database. The user has to pass parameters to the SQL Server that authenticates the user and provides access to the source table from which XML file is created. The user searches using the type ahead search which recommends words as the user types in Google. However this is a limited choice and may not cater to everyone's need. The system proposes to search data using Fuzzy that performs insertion/updating/deletion on the characters to generate a wide set of words and provide correctness in case of misspelled words.

Most searches checks for the values and identify their position inside XML file. However this system proposes to search either in value or tag or tag attribute or tag attribute value. During searching the assumptions are made such that the item being searched is unique, is ambiguous at other times. In direct search ambiguity does not occur and the user will have the details of the searched data. In the case of an ambiguity the via search mode is used.

In this mode the child and parent tag are provided to identify the searched data. The user can search on Tag, value, attribute and attribute value.

A facility to rank the data is extended in this paper. The xml file is ranked as follows, when an item is searched, records in xml containing the exact match is ordered first, followed by those matching the criteria plus other Example: when searching for people playing chess, people playing only chess are ranked above the people playing chess with other games, it is then followed by people not playing chess and then followed by people who do not have any sporting interest.

Existing System

TRADITIONAL methods use query languages such as XPath and XQuery to query XML data. These methods are powerful but unfriendly to non-expert users. First, these query languages are hard to comprehend for non-database users. In a traditional keyword-search system over XML data, a user composes a query, submits it to the system, and retrieves relevant answers from XML data. This information-access paradigm requires the user to have certain knowledge about the structure and content of the underlying data repository. In the case where the user has limited knowledge about the data, often the user feels "left in the dark" when issuing queries, and has to use a try-and-see approach for finding information.

Proposed System

In this paper, we propose TASX (pronounced "task"), a fuzzy type-ahead search method in XML data. TASX searches the XML data on the fly as user's type in query keywords, even in the presence of minor errors of their keywords. TASX provides a friendly interface for users to explore XML data, and can significantly save users typing effort. In this paper, we study research challenges that arise naturally in this computing paradigm. The main challenge is search efficiency. Each query with multiple keywords needs to be answered efficiently.

ARCHITECTURE

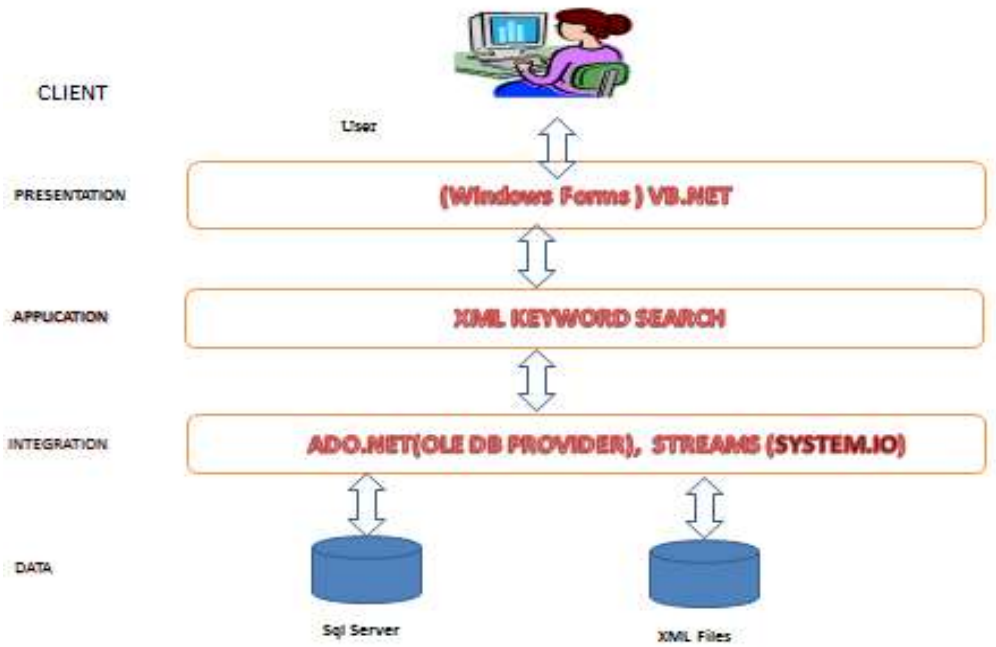

Fig 1- SOFTWARE ARCHITECTURE

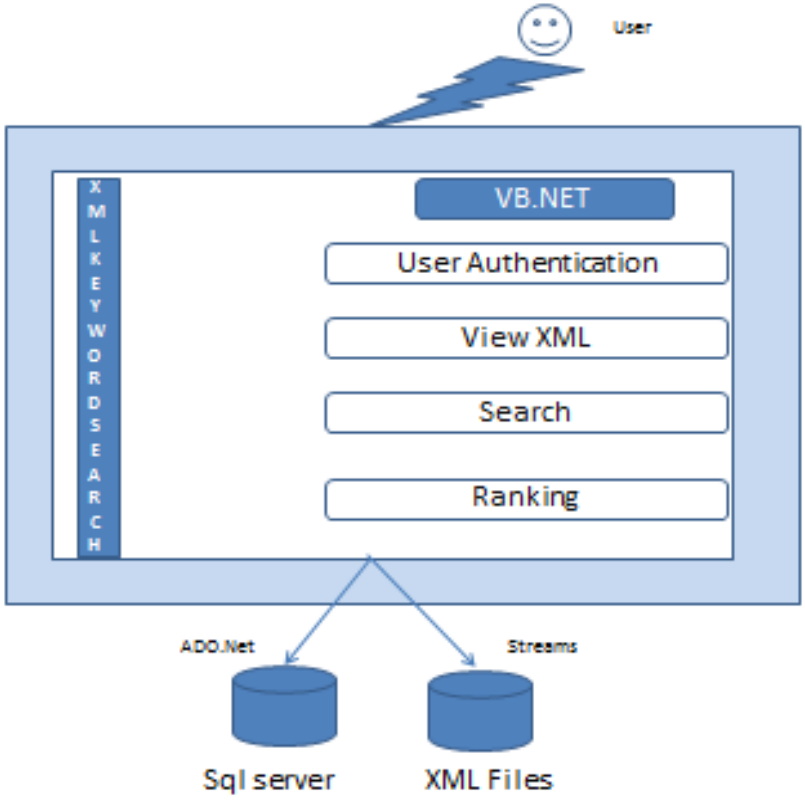

FIG 2. TECHNICAL ARCHITECTURE

SYSTEM ANALYSIS

Traditional methods use query languages such as XPath and XQuery to query XML data. These methods are powerful but unfriendly to no expert users. First, these query languages are hard to comprehend for no database users. For example, XQuery is fairly complicated to grasp.

Second, these languages require the queries to be posed against the underlying, sometimes complex, database schemas. Fortunately, keyword search is proposed as an alternative means for querying XML data, which is simple and yet familiar to most Internet users as it only requires the input of keywords. Keyword search is a widely accepted search paradigm for querying document systems and the World Wide Web. Recently, the database research community has been studying challenges related to keyword search in XML data One important advantage of keyword search is that it enables users to search information without knowing a complex query language such as XPath or XQuery, or having prior knowledge about the structure of the underlying data. In a traditional keyword-search system over XML data, a user composes a query, submits it to the system, and retrieves relevant answers from XML data. This information-access paradigm requires the user to have certain knowledge about the structure and content of the underlying data repository. In the case where the user has limited knowledge about the data, often the user feels "left in the dark" when issuing information. He tries a few possible keywords, goes through the returned results, modifies the keywords, and reissues a new query. He needs to repeat this step multiple times to find the information, if lucky enough. This search interface is neither efficient nor user friendly.

Many systems are introducing various features to solve this problem. One of the commonly used methods is Auto complete, which predicts a word or phrase that the user may type in based on the partial string the user has typed. More and more websites support this feature. As an example, almost all the major search engines nowadays automatically suggest possible keyword queries as a user types in partial keywords. Both Google Finance (http://finance.google.com/) and Yahoo! Finance (http://finance.yahoo.com/) support searching for stock information interactively as user's type in keywords. One limitation of Auto complete is that the system treats a query with multiple keywords as a single string; thus, it does not allow these keywords to appear at different places.

For instance, consider the search box on Apple.com, which allows Auto complete search on Apple products. Although a keyword query "iphone" can find a record "iphone has some great new features," a query with keywords "iphone features" cannot find this record (as of February 2010), because these two keywords appear at different places in the answer. To address this problem, Bast and Weber proposed Complete Search in textual documents, which can find relevant answers by allowing query keywords appear at any places in the answer. However, Complete-Search does not support approximate search that is it cannot allow minor errors between query keywords and answers. Recently, we studied fuzzy type-ahead search in textual documents. It allows users to explore data as they type, even in the presence of minor errors of their input keywords. Type-ahead search can provide users instant feedback as users type in keywords, and it does not require users to type in complete keywords. Type-ahead search can help users browse the data, save users typing effort, and efficiently find the information. We also studied type-ahead search in relational databases.

However, existing methods cannot search XML data in a type-ahead search manner, and it is not trivial to extend existing techniques to support fuzzy type-ahead search in XML data. This is because XML contains parent-child relationships, and we need to identify relevant XML sub trees that capture such structural relationships from XML data to answer keyword queries, instead of single documents. In this paper, we propose TASX (pronounced "task"), a fuzzy type-ahead search method in XML data. TASX searches the XML data on the fly as users type in query keywords, even in the presence of minor errors of their keywords. TASX provides a friendly interface for users to explore XML data, and can significantly save users typing effort. In this paper, we study research challenges that arise naturally in this computing paradigm. The main challenge is search efficiency.

Each query with multiple keywords needs to be answered efficiently. To make search really interactive, for each keystroke on the client browser, from the time the user presses the key to the time the results computed from the server are displayed on the browser, the delay should be as small as possible. An interactive speed requires this delay should be within milliseconds. Notice that this time includes the network transfer delay, execution time on the server, and the time for the browser to execute its Java- Script. This low-running-time requirement is especially challenging when the backend repository has a large amount of data. To achieve our goal, we propose effective index structures and algorithms to answer keyword queries in XML data. We examine effective ranking functions and early termination techniques to progressively identify top-k answers. To the best of our knowledge, this is the first paper to study fuzzy type-ahead search in XML data. To summarize, we make the following contributions: . We formalize the problem of fuzzy type-ahead search in XML data. . We propose effective index structures and efficient algorithms to achieve a high interactive speed for fuzzy type-ahead search in XML data.

SYSTEM DESIGN

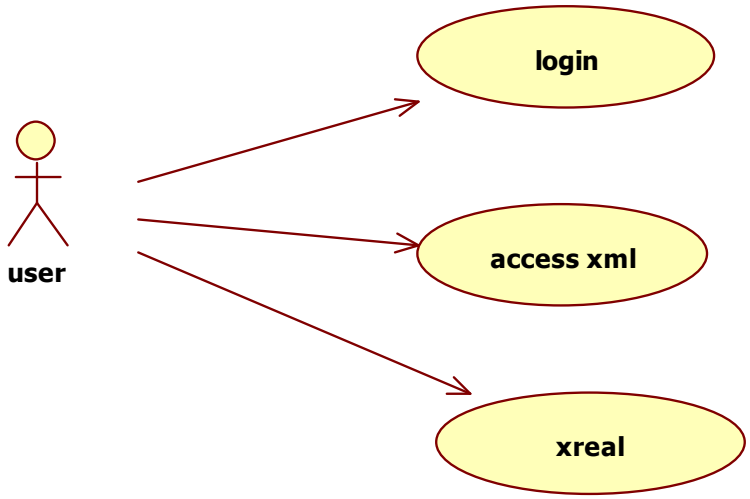

Fig 3. Use Case for Login

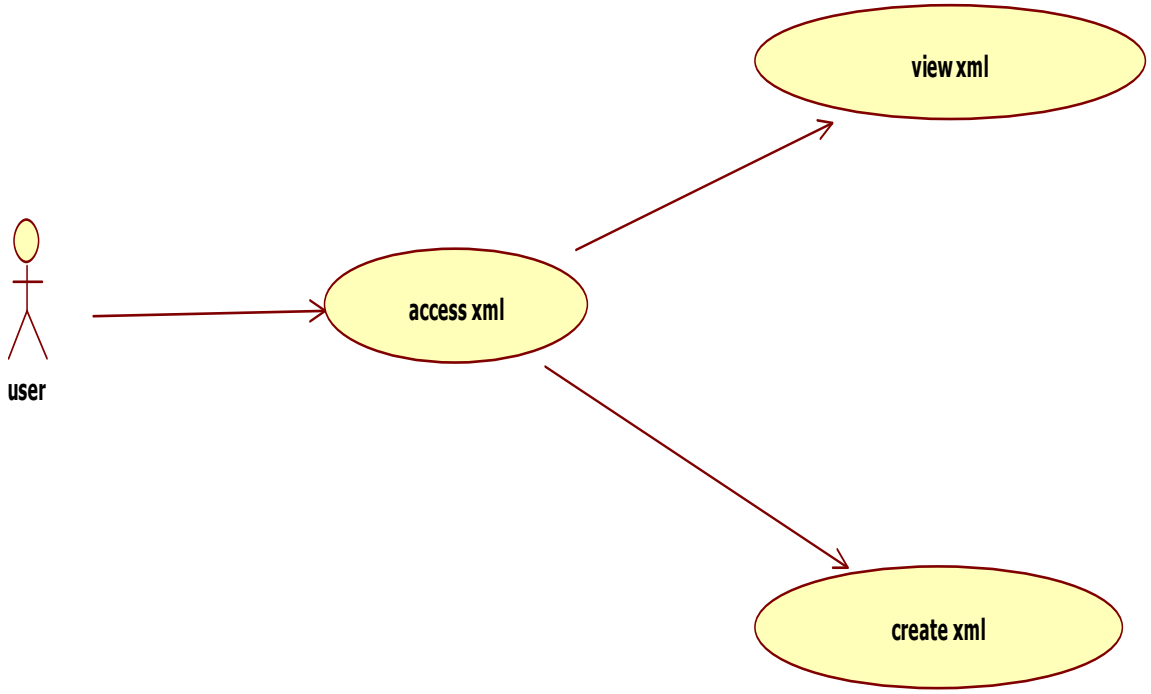

Fig 3.1. Use Case for Accessing XML

CLASS DIAGRAM

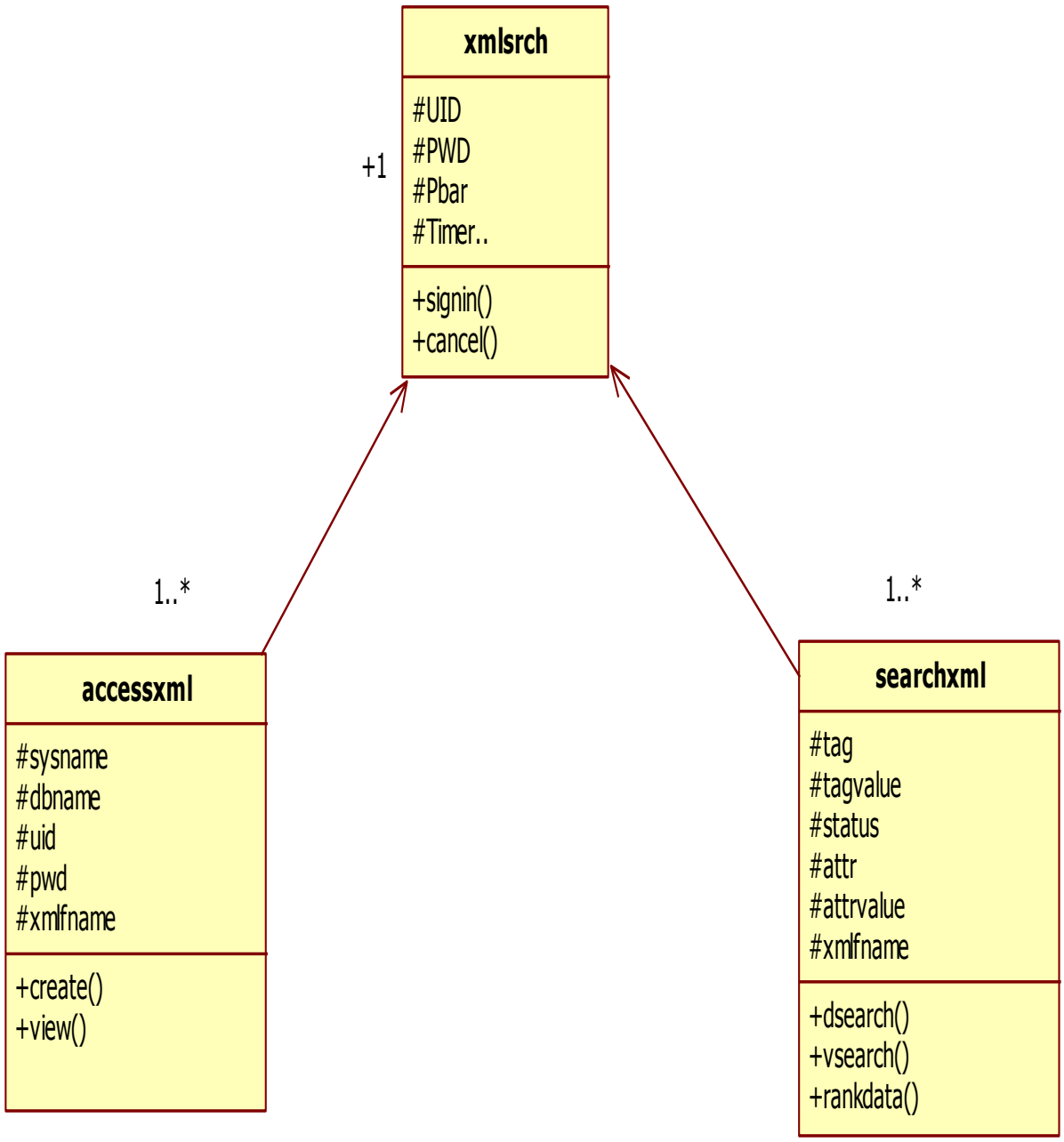

Fig 4. Class Diagram

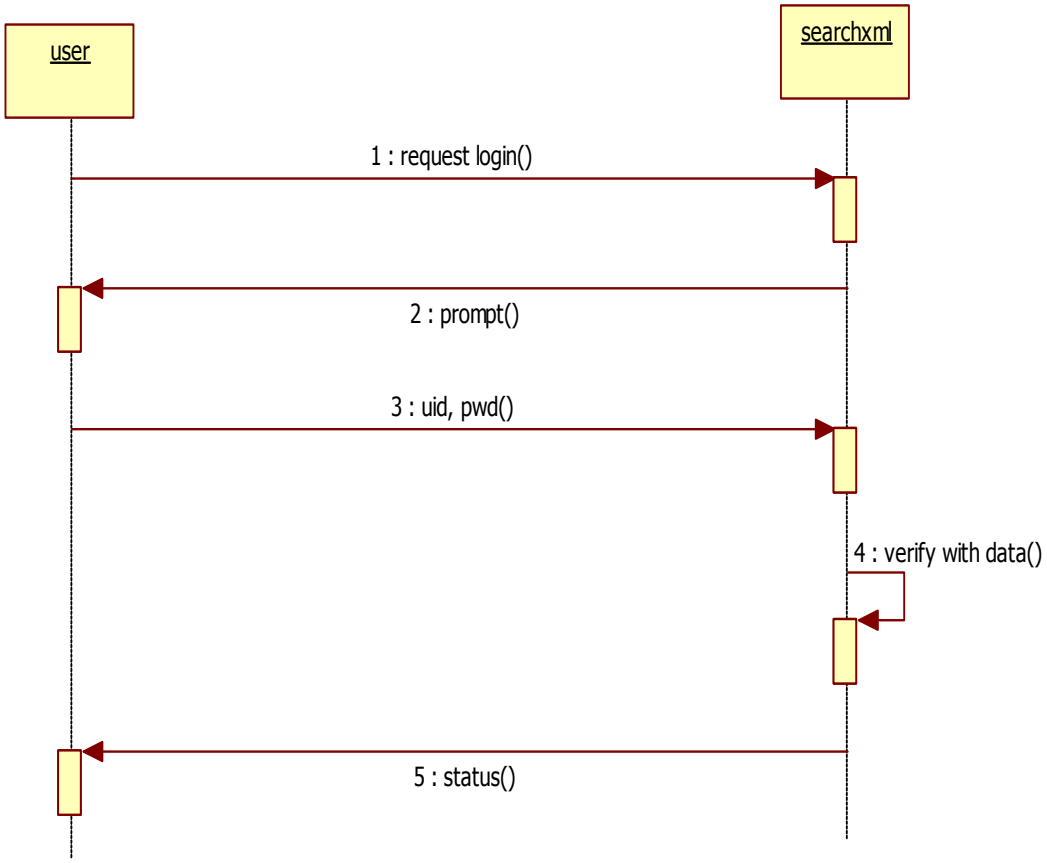

FIG 5.CLASS DIAGRAM

CONCLUSION AND FUTURE WORK

In this paper, the study of the problem of effective XML keyword search which includes the identification of user search intention and result ranking in the presence of keyword ambiguities is done. I utilized statistics to infer user search intention and rank the query results. In particular, defined XML TF and XML DF, based on which we design formulae to compute the confidence level of each candidate node type to be a *search for/search via node*, and further propose a novel *XML TF*IDF similarity* ranking scheme to capture the hierarchical structure of XML data. Lastly, the popularity of a query result (captured by IDRef relationships) is considered to handle the case that multiple results have comparable relevance scores. In future, I would like to extend our approach to handle the XML document conforming to a highly recursive schema as well.

Future Scope

- THE APPLICATION CAN BE EXTENDED TO SEARCH ON WEB.
- FACILITY TO INCORPORATE DOWNLOADING OF FILES.
- CAN BE EXTENDED TO SEARCH IN MULTIPLE XML FILES.

REFERENCES

[1] S. Agrawal, S. Chaudhuri, and G. Das, "Dbxplorer: A System for Keyword-Based Search over Relational Databases," Proc. Int'l Conf. Data Eng. (ICDE), pp. 5-16, 2002.

[2] S. Amer-Yahia, D. Hiemstra, T. Roelleke, D. Srivastava, and G.Weikum, "Db&ir Integration: Report on the Dagstuhl Seminar 'Ranked Xml Querying'," SIGMOD Record, vol. 37, no. 3, pp. 46-49, 2008.

[3] M.D. Atkinson, J.-R. Sack, N. Santoro, and T. Strothotte, "Min-max Heaps and Generalized Priority Queues," Comm. ACM, vol. 29, no. 10, pp. 996-1000, 1986.

[4] A. Balmin, V. Hristidis, and Y. Papakonstantinou, "Objectrank:Authority-Based Keyword Search in Databases," Proc. Int'l Conf.Very Large Data Bases (VLDB), pp. 564-575, 2004.