EFFICIENT TYPE-AHEAD SEARCH IN XML DATA

1 SYED. AMZAD, 2SYED. AKHTAR BASHA, 3P.BABU, 4 R.PRAVEEN SAM

1PG Scholar QCET Nellore, 2,3Associate Professor, 4Professor

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.

1. 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.

2. 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 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.

3. PROBLEM FORMULATION OF FUZZY TYPE-AHEAD SEARCH IN XML DATA

In this section, we introduce the overview of fuzzy typeahead search in XML data and formalize the problem.

3.1 Overview

We first introduce how TASX works for queries with multiple keywords in XML data, by allowing minor errors of query keywords and inconsistencies in the data itself. Assume there is an underlying XML document that resides on a server. A user accesses and searches the data through a web browser. Each keystroke that the user types invokes a query, which includes the current string the user has typed in. The browser sends the query to the server, which computes and returns to the user the best answers ranked by their relevancy to the query.

The server first tokenizes the query string into several keywords using delimiters such as the space character. The keywords are assumed as partial keywords, as the user may have not finished typing the complete keywords. For the partial keywords, we would like to know the possible words the user intends to type. However, given the limited information, we can only identify a set of complete words in the data set which have similar prefixes with the partial keywords. This set of complete words are called the predicted words. We use edit distance to quantify the similarity between two words. The edit distance between two words s1 and s2, denoted by edðs1; s2Þ, is the minimum number of edit operations (i.e., insertion, deletion, and substitution) of single characters needed to transform the first one to the second. For example, edðmics; nicesÞ ¼ 1 and edðmics; michÞ ¼ 1. For instance, given a partial keyword “mics,” its predicted words could be “mices,” “mich,” “michal,” etc. Then, the server identifies the relevant subtrees in XML data that contain the predicted words for every input keyword. We can use any existing semantics to identify the answer based on the predicted words, such as ELCA. We call these relevant subtrees the predicted answers of the query. For example, consider the XML document in Fig. 1. Assume a user types in a keyword query “db mics.” The predicted word of “db” is “db.” The predicted words of “mics” are “mices” and “mich.” The subtree rooted at node 12 is the predicted answer of “db mices.” The subtree rooted at node 15 is the predicted answer of “db mich.” Thus, TASX can save users time and efforts, since they can find the answers even if they have not seen approach for finding information.
4. PROGRESSIVE AND EFFECTIVE TOP-K FUZZY TYPE-AHEAD SEARCH

The LCA-based fuzzy type-ahead search algorithm in XML data has two main limitations. First, they use the “AND” semantics between input keywords of a query, and ignore the answers that contain some of the query keywords (but not all the keywords). For example, suppose a user types in a keyword query “DB IR” on the XML document in Fig. 1. For the keyword “IR,” we index nodes 6, 16, 24, 5, 15, 23, 2, 20, and 1. For the keyword “DB,” we index nodes 13, 16, 12, 15, 9, 17, 12, 15, 9, 2, 8, 1, and 5. The nodes are sorted by their relevance to the keyword (we will discuss how to evaluate relevance of nodes to a keyword in Section 5.2.1). Fig. 3 gives the extended trie structure. For instance, assume a user types in a keyword query “DB IR Tom.” We use the extended trie structure to find nodes 15 and 12 as the top-2 relevant nodes. We propose minimal-cost trees (MCTs) to construct the best tree structure to address the second challenge.

CONCLUSIONS AND FUTURE WORK

In this paper, we studied the problem of fuzzy type-ahead search in XML data. We proposed effective index structures, efficient algorithms, and novel optimization techniques to progressively and efficiently identify the top-k answers. We examined the LCA-based method to interactively identify the predicted answers. We have developed a minimal-cost-tree-based search method to efficiently and progressively identify the most relevant answers. We proposed a heap-based method to avoid constructing union lists on the fly. We devised a forward-index structure to further improve search performance. We have implemented our method, and the experimental results show that our method achieves high search efficiency and result quality.
REFERENCES


***