International Journal of Emerging Trend in Engineering and Basic Sciences (IJEEBS)
ISSN (Online) 2349-6967
Volume 2 , Issue 1(Jan-Feb 2015), PP158-162

Proximity Ranking Based Fast Fuzzy Search under Real Time Environment: A Survey

Shraddha V. Khandade, Prof. Sandeep U. Kadam
Department of Computer engineering, Ambi, Pune, India

Abstract :- In conventional keyword-search system over XML data user composes a keyword query and submits to the system. When the user has a limited knowledge about the data, he feels confused when issuing queries. He has to use a trial and error method for finding out the needed information. Then the fuzzy type search in XML data is examined for avoiding constant problems in search. It is a new information access paradigm in the system which searches XML data when the user types in different query keywords. It allows users to explore the data as they type when the minor errors are present in their keywords. This method has four features: 1) this is Auto complete and it supports queries with multiple keywords in XML data. 2) Fuzzy search can find high quality answers that have keywords matching the query keywords approximately. 3) The index structures and searching algorithms in this method are achieving a very high interactive speed. The effective index structures and top-k algorithms are achieving a high interactive speed. The effective ranking functions are studied progressively to identify the top-k relevant answers. 4) Proximity aware ranking improves precision of top results significance and quality.

Keywords :- Fuzzy search, Keyword search, Proximity ranking, Top-k answer, XML.

I. INTRODUCTION

Users of search services care about both result quality and retrieval time. But balancing efficiency and effectiveness becomes increasingly difficult, as collection size grows. Although the top ranked results returned by existing information retrieval models and can satisfy a user's basic requirements, many weakly relevant or no relevant documents are also returned. This occurs because many relevance models only consider a bag-of-words representation of documents, without taking into account the locations within the document at which those matching words occur. That is, bag-of-word retrieval models do not allow for the intuition that if the query terms occur near each other in a document, it may indicate that document is more relevant. Fuzzy search also used in XML data. 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 non database users. Second, these languages require the queries to be posed against the underlying, sometimes complex, database schemas. 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. 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. In proposed work we are using fuzzy search system in our information access paradigm.

In this TASX (pronounced "task"),[1] a fuzzy type-ahead search method in XML data, is used. 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.

Exact Search: It is first considered the case of exact search. One naive way to process such a query on the server is to answer the query from scratch as follows: first find the trie node corresponding to this keyword by traversing the trie from the root. Then, we locate the leaf descendants of this node, and retrieve the corresponding predicted words and the predicted XML elements on the inverted lists.

Fuzzy Search: Users often make some mistakes in their search queries. Meanwhile, small sized keyboards on mobile devices, lack of caution, or limited knowledge about the data can also cause mistakes. In this case we cannot find relevant answers by finding records with keywords matching the query exactly. This problem can be solved by supporting fuzzy search.

Finding Relevant Answers within Time Limit: A main computational challenge in this search paradigm is its high- speed requirement. It is known that to achieve an instant speed for humans (i.e., users do not feel delay), from the time a user types in a character to the time the results are shown on the device, the total time should be within 100 milliseconds . The time includes the network delay, the time on the search server, and the time of running code on the device of the user (such as JavaScript in browsers). Thus the amount of time the server can spend is even less. At the same time, compared to traditional search systems, instant search can result in more queries on the server since each keystroke can invoke a query, thus it requires a higher speed of the search process to meet the requirement of a high query throughput. What makes the computation even more challenging is that the server also needs to retrieve high-quality answers to a query given a limited amount of time to meet the information need of the user.

Proximity Ranking: Recent studies show proximity is highly correlated with document relevancy, and proximity aware ranking improves the precision of top results significantly. However, there are only a few studies that improve the query efficiency of proximity-aware search by using early-termination techniques. Zhu et al. [2] exploited document structure to build a multi-tiered index to terminate the search process without processing all the tiers. The techniques create an additional inverted index for all term pairs, resulting in a large space. To reduce the index size, Zhu et al. Proposed to build a compact phrase index for a subset of the phrases.

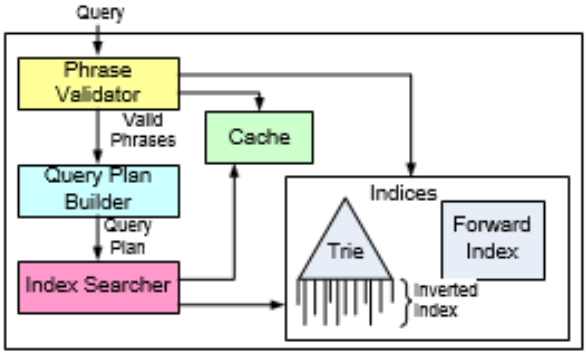

Fig.1. Server architecture of instant fuzzy search.

Subsequent queries of the user typically share many keywords with previous queries due to incremental typing, it is very important to do the computation incrementally and distribute the computational cost of a query between its preceding queries. For this reason, we have a Cache module that stores some of the computed results of early queries that can be used to expedite the computation of later queries. The Phrase Validator uses the Cache module to validate a phrase without traversing the trie from scratch, while the Index Searcher benefits from the Cache by being able to retrieve the answers to an earlier query to reduce the computational cost.

II. METHODS FOR KEYWORD SEARCH OVER XML DATA

a. LCA based method:

The lowest common ancestor (LCA) is a concept in graph theory and computer science. Let T be a rooted tree with n nodes. The lowest common ancestor between two nodes v and w is defined as the lowest node in T that has both v and w as descendants. The LCA of v and w in T is the shared ancestor of v and w that is located farthest from the root. There are different ways to answer the query on an xml document; one commonly used method is LCA based method [3]. Many algorithms that use query over xml uses this method. Content nodes are the parent node of the keyword. For example consider keyword db in fig.2 then content node of db is node 13 and node16.The server contains index structure of xml document which each node is letter in keyword and leaf node contain all nodes that contain the keyword this leaf node is called inverted list.

Procedure-

- For keyword query the LCA based method retrieves content nodes in xml that are in inverted lists.
- Identify the LCAs of content nodes in inverted list

- Takes the sub tree rooted at LCAs as answer to the query
  for example suppose the user typed the query "www db" then the content nodes of db are{13,16} and for www are3 ,the LCAs of these content nodes are nodes ,12,15,2,1.here the nodes 3,13,12,15 are more relevant answers but nodes 2 and 1 are not relevant answers.

Limitation-

- It gives irrelevant answers
- The results are not of high quality

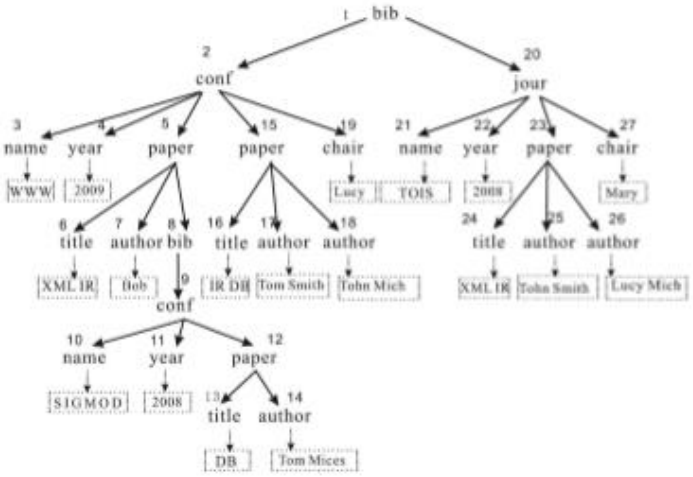

Fig. 2 An XML document.

Consider the XML document in Fig.2. 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 finished typing all the complete keywords or typing keywords with minor errors.

b. ELCA based method:

To address the limitation of LCA based method exclusive LCA (ELCA) [4] is proposed. It states that an LCA is ELCA if it is still an LCA after excluding its LCA descendents. for example suppose the user typed the query "db tom" then the content nodes of db are {13, 16} and for tom are{14.17} ,the LCAs of these content nodes are nodes2,12,15,1.here the ELCAs are 12,15.the subtree rooted with these nodes is displayed which are relevant answers Node 2 is not an ELCA as it is not an LCA after excluding nodes 12 and 15. Xu and Papakonstantinou proposed a binary-search-based method to efficiently identify ELCAs.

III. LITERATURE SURVEY

Keyword search is a widely accepted search paradigm for querying document systems and the World Wide Web.

L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram proposed challenges related to keyword search in XML data in paper "Xrank: Ranked Keyword Search over Xml Documents." [5] 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. Fuzzy type-ahead search in textual documents.

S. Ji, G. Li, C. Li, and J. Feng, in paper "Efficient Interactive Fuzzy Keyword Search" [6] 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. Type-ahead search in relational databases also studied in. There are studies on building an additional index for each term pair that appears close to each other in the data, or for phrases However, building an index for the term pairs will consume a significant amount of space. Studies show that users often include entities such as people names, companies, and locations in their queries. These entities can contain multiple keywords, and the user wants these keywords to appear in the answers as they are, i.e., the keywords are adjacent and in the same order in the answers as in the query. Users sometimes enter keywords enclosed by quotation marks to express that they want those keywords to be treated as phrases. Based on this observation, we propose a technique that focuses on the important case where we rank highly those answers containing the query keywords as they are, in addition to adapting existing solutions to instant-fuzzy search

Jianhua Feng, and Guoliang Li" Efficient Fuzzy Type-Ahead Search in XML Data".[1] 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.

Y. Xu and Y. Papakonstantinou, "Efficient LCA Based Keyword Search in XML Data"[4] proposed an efficient algorithm called Indexed Stack to find answers to keyword queries based on XRank's semantics to LCA. Also focused on three most closely related works: XRank, schema free Xquery and XK- search.

.

IV. PROPOSED WORK

There are many works to do in the future. Firstly, users reported that the returned local results of Title input box cannot provide useful information for faceted search because there are little publications/movies that have the same title. We should summarize these local results into groups to improve the faceted search ability. Secondly, our systems support only AND-semantics for the queries. We will investigate the OR-semantics and top-k algorithms in the future to make the searching more flexible and efficient. Thirdly, we should also tolerate misplacing of keywords to leverage the simplicity and usability of forms.

By this comparative study, It is analysed that search-as-you-type can be further enhanced. This can be achieved by using ranking queries. Reference points, called as vantage points are used to partition the relational data space into spherical shell-like regions in a hierarchical manner. These vantage points are distance-based index structures. Usage large number of vantage points may result in a more efficient result during search operations.

V. CONCLUSION

In this it is studied how to improve ranking of an instant-fuzzy search system by considering proximity information when we need to compute top-k answers and how to adapt existing solutions to solve this problem, including computing all answers, doing early termination, and indexing term pairs. A technique is proposed to index important phrases to avoid the large space overhead of indexing all word grams. An incremental computation algorithm is presented for finding the indexed phrases in a query efficiently, and studied how to compute and rank the segmentations consisting of the indexed phrases. The techniques are compared to the instant fuzzy adaptations of basic approaches. A very thorough analysis is conducted by considering space, time, and relevancy tradeoffs of these approaches. In particular, the experiments on real data showed the efficiency of

the proposed technique for keyword queries that are common in search applications. It is concluded that computing all the answers for the other queries would give the best performance and satisfy the high efficiency requirement of instant search.

REFERENCES

[1] L. Jianhua Feng, and Guoliang Li,“ Efficient Fuzzy Type-Ahead Search in XML Data”, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 24, NO. 5, MAY 2012
[2] M. Zhu, S. Shi, M. Li, and J.-R. Wen, “Effective top-k computation in retrieving structured documents with term-proximity support,” in CIKM, 2007, pp. 771–78.
[3] Y. Xu and Y. Papakonstantinou, “Efficient Keyword Search for Smallest Lcas in XML Databases,” Proc. ACM SIGMOD Int’l Conf. Management of Data, pp. 537-538, 2005.
[4] Y. Xu and Y. Papakonstantinou, “Efficient LCA Based Keyword Search in XML Data,” Proc. Int’l Conf. Extending Database Technology: Advances in Database Technology (EDBT), pp. 535-546, 2008.
[5] F. Shao, L. Guo, C. Botev, A. Bhaskar, M.M.M. Chettiar, F.Y. 0002, and J. Shanmugasundaram, “Efficient Keyword Search over Virtual XML Views,” Proc. Int’l Conf. Very Large Data Bases (VLDB), pp. 1057-1068, 2007
[6] S. Ji, G. Li, C. Li, and J. Feng, “Efficient Interactive Fuzzy Keyword Search,” Proc. Int’l Conf. World Wide Web (WWW), pp. 371-380, 2009