Volume 5, Issue 11, November 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Proximity Ranking Based Fast Fuzzy Search: A Survey

Shraddha Khandade
Dpartment of Computer Engineering,
Dr. D. Y. Patil College of Engineering, Pune,
Maharashtra, India

Abstract*— Instant search is an information-retrieval paradigm in that,a system finds answers to a query instantly when a user types in keywords character-by-character. Fuzzy search further improves user search experiences because it finds relevant answers with keywords similar to query keywords. The main challenge in this paradigm is high speed requirement i.e. this paradigm needs to be answered to query in milliseconds.In this paper, we study how to integrate proximity information into ranking in instant-fuzzy search while achieving efficient time and space complexities. A naıve solution is to compute all answers then to rank them, butwhen there are too many answers, it cannot meet this high speed requirement on large data sets, so there are studies of early termination techniques to efficiently compute relevant answers. To overcome the space and time limitations of these solutions, we propose an approach that focuses on common phrases in the data and queries, assuming records with these phrases are ranked higher. We study how to index these phrases and develop an incremental computation algorithm for efficiently segmenting a query into phrases and computing relevant answers.

Keywords*— Fuzzy search, XML data, Instant search, Proximity ranking, Keyword search.

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 can satisfy a user's basic requirements, many weakly relevant or nonrelevant 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. Traditional methods use query languages such as XPath and XQuery to query XML data. These methods are powerful but unfriendly to nonexpert users. First, these query languages are hard to comprehend for nondatabase 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 knowledgeaboutthestructureandcontentoftheunderlying 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.

Instant Search: An instant search returns the answers immediately based on a partial query a user has typed in. For example, the Internet Movie Database, IMDB, has a search interface that offers instant results to users while they are typing queries1. When a user types in "sere", the system returns answers such as "Serena", "Serenity", "Serendipity", and "Serena Williams". Many users prefer the experience of seeing the search results instantly and formulating their queries accordingly instead of being left in the dark until they hit the search button. Our recent study showed that this new information-retrieval method helps users find their answers quickly with less effort.

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.

Proximity Ranking: Proximity measures the closeness or nearness of a keywords. The ranking to searched result is given depending on proximity. Proximity ranking will rank documents higher where the query words are found *close together .*

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.

II. 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."[1]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 [2]S. Ji, G. Li, C. Li, and J. Feng,in paper "Efficient Interactive Fuzzy Keyword Search" 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 instantfeedback 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 [2].

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. For instance, the approach reported an index of 1.3 TB for a collection of 25 million documents, and reduced the size to 343.5 GB by pruning the lists horizontally. In addition, these studies focus on two-keyword queries only, and do not consider queries with more keywords.
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

In paper," Efficient Fuzzy Type-Ahead Search in XML Data" JianhuaFeng, and Guoliang Li [3] proposed 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.

Chen Li, Jiaheng Lu, Yiming Lu has explained how to efficiently find in a collection of strings those similar to a given string. The existing methodology used individual filters for approximate search. The author has developed new algorithms that can greatly improve the performance of existing algorithms. The authors C. Li, J. Lu, and Y. Lu in paper, "Efficient Merging and Filtering Algorithms for Approximate String Searches" has explained how to integrate existing filtering techniques with the algorithms, and showed that they should be used together judiciously, since the way to do the integration can greatly affect the performance.[4].

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 subtrees that capture such structural relationships from XML data to answer keyword queries, instead of single documents.

A. Scalability

In the paper "Search-As-You-Type in Forms: Leveraging the Usability and the Functionality of Search Paradigm in Relational Databases" author Hao Wu proposed that both of the index size and the response time increase linearly as the dataset gets larger (see Section 3.2). When the size of the dataset is too large, we cannot store the index in the main memory. In this case, it is necessary to investigate other ways to build the index, as well as more scalable search algorithms. We investigate two of the possible solutions to this issue: (1) use native DBMS support and (2) design a top-k algorithm.[5].

Native DBMS Support:

When data is stored in a relational database, it is natural to use SQLs to perform the searching. In this way, the system would require less efforts to deploy and less main memory to run. In addition, we could also pay less attention on reducing the index size, since the storage and retrieval algorithms are already carefully designed and chosen in the system natively. That is to say, we only put them all in the DBMS, and then use DBMS capabilities to support the two new features. Nevertheless, how to maintain the data and the indexes, as well as how to design a high-performance search algorithm, are still need to be take into consideration. As described before, the whole search process can be divided to three sub-tasks: (1) search for local results, (2) search for global results, and (3) synchronization.

Top-k Algorithms:

The algorithm incorporated by our initial work can hardly be adapted to handle large datasets, since it is needed to achieve high interactive speed. Traditional keyword search algorithms incorporate top-k algorithms to guarantee that the

query time increases sub-linearly as dataset grows. With a top-k search algorithm, given a user defined parameter k, the system returns only k top-ranked results, instead of all of them to the user.

B. Answering Queries with Multiple Keywords

Now, we consider how to do fuzzy type-ahead search in the case of a query with multiple keywords. For a keystroke that invokes a query, we first tokenize the query string into keywords, $k_1,k_2,...,k_l$ '. For each keyword k_i ($1 \leq i \leq l$) we compute its corresponding active nodes, and for each such active node, we retrieve its leaf descendants and corresponding inverted lists. Then, we compute union list for every $\bar{U}_{ki}$ Finally, the predicted answers are computed on top of lists $\bar{U}_{k1},\bar{U}_{k2},...,\bar{U}_{kl}$'. We use the semantics of ELCA [6] to compute the corresponding answers. The binary-search-based method is used to compute ELCAs [6].

Type-ahead search is a new topic to query relational databases. Li et al. [7] studied type-ahead search in relational databases, which allows searching on the under- lying relational databases on the fly as users type in query keywords. Ji et al. [2] studied fuzzy type-ahead search on a set of tuples/documents, which can on the fly find relevant answers by allowing minor errors between input keywords andtheunderlyingdata.Astraightforwardmethodfortype- ahead search in XML data is to first find all predicted words, and then use existing search semantics, e.g., LCA and ELCA, to compute relevant answers based on the predicted words. However, this method is very time consuming for finding top-k answers. To address this problem, we propose to progressively find the most relevant answers. For exact search, we propose to incrementally compute predicted words. For fuzzy search, we use existing techniques [2] to compute predictedwords of querykeywords. We extend the ranking functions in [7] to support fuzzy search, and propose new index structures and efficient algorithms to progressively find the most relevant answers.

III. 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.

IV. CONCLUSION

In this paper 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. Thetechniques 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. Guo, F. Shao, C. Botev, and J. Shanmugasundaram, "Xrank: Ranked Keyword Search over Xml Documents,"Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 16-27, 2003.

[2] 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

[3] JianhuaFeng, Senior Member, IEEE, and Guoliang Li, Member, IEEE "Efficient Fuzzy Type-Ahead Search in XML Data"

[4] C. Li, J. Lu, and Y. Lu, "Efficient Merging and Filtering Algorithms for Approximate String Searches," Proc. IEEE 24th Int'l Conf. Data Eng. (ICDE '08), pp. 257-266, 2008

[5] KarthigaChandrasekaran, Karpagalakshmi.R.C "A Survey on Search-As-You-Type Using Ranking Queries"

[6] 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.

[7] G. Li, S. Ji, C. Li, and J. Feng, "Efficient Type-Ahead Search on Relational Data: A Tastier Approach," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 695-706, 2009.

[8] Cetindil, J. Esmaelnezhad, C. Li, and D. Newman, "Analysis of instant search query logs," in WebDB, 2012, pp.7–12.

[9] R. B. Miller, "Response time in man-computer conversational transactions," in Proceedings of the December 9-11, 1968, fall joint computer conference, part I, ser. AFIPS '68 (Fall, part I). New York, NY, USA: ACM, 1968, pp. 267–277. [Online]. Available: http://doi.acm.org/10.1145/1476589.1476628

[10] C. Silverstein, M. R. Henzinger, H. Marais, and M. Moricz, "Analysis of a very large web search engine query log," SIGIR Forum, vol. 33, no. 1, pp. 6–12, 1999.

[11] G. Li, J. Wang, C. Li, and J. Feng, "Supporting efficient top-k queries in type-ahead search," in SIGIR, 2012, pp. 355–364.

[13] H. Yan, S. Shi, F. Zhang, T. Suel, and J.-R.Wen, "Efficient term proximity search with term-pair indexes," in CIKM, 2010, pp. 1229– 1238.[7] M. Zhu, S. Shi, N. Yu, and J.-R. Wen, "Can phrase indexing help to process non-phrase queries?" in CIKM, 2008, pp. 679–688.

