International Journal of Engineering Research and General Science Volume 3, Issue 2, March-April, 2015
ISSN 2091-2730

A Survey:A Flexible Approach to Instant-Fuzzy Search Using Effective Phrase Indexing and Segmentation with Proximity Rankin

Ramesh S. Yevale[1]
Dept. of Computer Engineering,
ICOER,Wagholi.
Pune, India
ryevale33@gmail.com

Prof.Vinod S. Wadne[2]
Dept. of Computer Engineering,
ICOER,Wagholi
Pune,India
vinods1111@gmail.com

Abstract—Instant search is basically important task for the user to get effective responses to the query when user typing a query in search engine. Fuzzy search used to improve user search experiences by finding relevant answers with keywords similar to query keywords.We are using phrase threshold value which is used to limit the answer set generated by instant fuzzy search. For that main challenge is that to improve the speed of performance as well as minimize answer set to retrieval of desired documents for the user query. At the same time, we also need better ranking functions that consider the proximity of keywords to compute relevance scores. In this paper, we study how to compute proximity information into ranking in instant-fuzzy search while achieving efficient time and space complexities. A phrase base indexing technique is used to overcome the space and time limitations of these solutions, we propose an approach that focuses on common phrases in the database. We study how to index these phrase threshold value and compare user threshold for effective answer set and develop an computational algorithm for efficiently segmenting a query into phrases and computing these phrases using algorithm to find relevant answers to the user query.

Keywords—instant search,fuzzy search,phrase threshold

INTRODUCTION

Generally, instant search is an important in Information Retrieval(IR) for user to get fast response from search engine when user typing an query.[1]Instant search is said to be effective when it gives faster retrieval of answer set with minimum computational time. It is known that to achieve an instant speed for humans, from the time a user types in a character to the time the results are shown in answer set, the total time should be less than 100 milliseconds [7]. It is needed to consider time goes in network delay, time on the search server to find relevant documents to the query, and the time of running code on the device of the user such as web browser. 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.[8]

Basically, text related interfaces have been undergoing a sea change in the last few years.An autocompletion mechanism unobtrusively prompts the user with a set of suggestions, each of which is a suffix, or completion, of the user's current input. This allows the user to avoid unnecessary typing, hence saving not just time but also user cognitive burden.[2]

Especially, management of string data in databases and information systems increased huge importance in current time.If suppose, given a collection of strings, efficiently identify the ones similar to a given query string. Such a query is called an "approximate string search." This problem is of great interest for a variety of applications, as illustrated by the following examples.

Generally,information from multiple data sources of ten have numerous inconsistencies as data may be present in different formats. For example, the same real-world entity can be represented in slightly different formats, such as "PO Box 13, Main St." and "P.O. Box 13, Main St". Errors can also be introduced due to irregularities in the data collection process, from human mistakes, and many other causes. For these reasons, one of the main goals of data cleaning is to find similar entities within a collection, or all similar pairs of entities across a number of collections.

Users may pose SQL queries to a DBMS that contain selection predicates that do not match all of the relevant data within the database exactly. The reasons are possible errors in the query, may be inconsistencies in the data, limited knowledge about the data, and more. By supporting query relaxation, the DBMS can return data of potential interest to the user, based on query predicate similarity (e.g., returning "Steve Smith" as an answer to the query "Steven Smith").

It is needed to retrieve the relevant answer set to user query while user is typing in a search engine (e.g.,Google or Yahoo search box with a drop-down suggestion menu that updates as users type). These interactive search boxes are ubiquitous and have shown to be very important in practice, because they limit the number of errors made by users and also reduce the number of query reformulations submitted in order to find the one that will yield satisfying results to the user. The drawback of almost all existing, interactive techniques is that they support only prefix or substring matches, without regard for fuzzy, approximate searching; if users make a spelling mistake, they are presented with an empty suggestion box. One reason is that interactive approximate string search has attracted little attention and is not a trivial problem to solve, given the expensive nature of string similarity functions and ranking techniques.[3]

Problem statement :

In this paper, we study how to integrate proximity information into ranking in instant-fuzzy search to compute relevant answers to the query[8]. The proximity of matching keywords in answers is an important function to determine the relevance of the answers. User queries typically contain correlated keywords, as well as some pattern based phrases and to answers to these keywords or phrases together are more likely what the user is looking for. [15]

For example, if the user types in a search engine as a search query "Sachin Tendulkar", the user is most likely looking for the records containing information about the cricketer Sachin Tendulkar, while documents containing "Sachin Pilgaonkar".Existing system have limitation to respond to user query,as it requires mostly three phrases to enter for proximity instant search and it is time consuming.To achieve exact matches needed to user,we adapt instant fuzzy search.There is a need to minimize time and space tradeoff for retrieval of user query while user typing a query.

LITEATURE SURVEY

Significant works have been done in instant search in which sytsem finds answer to query while user types in keyword character-by-character.Some researches defined the techniques to integrate proximity information into ranking in instant fuzzy search.

In [1], a Proximity Probabilistic Model (PPM) that uses and advances a bag-of-words probabilistic retrieval model. In this paper, a document is transformed to a pseudo document form, in which a term count is propagated to other nearby terms. Then they consider three heuristics, i.e., the distance of two query term occurrences, their order, and aasigned term weights, which can be viewed as a pseudo term frequency. Finally, integrate term proximity into the probabilistic model BM25 by using the pseudo term frequency to replace term frequency.

In [2], author studied the problem of autocompletion at the level of a multi-word "phrase" which is appeared in a query. There are two main challenges: one is that the number of phrases (both the number possible as well as the number actually observed in a corpus) is combinatorially larger than the number of words; the second is that a "phrase", unlike a "word", does not have a well-defined boundary, so that the autocompletion system has to decide not just what to predict, but also how far way these phrases indeed.For that implementation they introduced a FussyTree structure to address the first challenge and the concept of a significant phrase to address the second challenge.They developed a probabilistically driven multiple completion choice model, and exploit features such as frequency distributions to improve the quality of our suffix completions.They experimentally demonstrate the practicability and value of our technique for an email composition application and show that we can save approximately a fifth of the keystrokes typed.

In [3], author studied hoe to provide a comprehensive overview of recent research progress on the important problem of approximate search in string collections. We identify existing indexes, search algorithms, filtering strategies, selectivity estimation techniques and other work, and comment on their respective merits and limitations.

In [4], author present the first large-scale study of user in- teractions with auto-completion based on query logs of Bing, a commercial search engine. Their results confirm that lower-ranked auto-completion suggestions receive substantially lower engagement than those ranked higher. They also observe that users are most likely to engage with auto-completion after typing about half of the query, and in particular at word boundaries. Author noticed that the likelihood of using auto-completion varies with the distance of query characters on the keyboard.Finally, the results reported in their study provide valuable insights for understanding user engagement with autocompletion, and are likely to inform the design of more effective QAC systems.

In [5], author implemented a novel and probabilistic approach to string transformation, which is both accurate and efficient for the retrieval of answer set. This approach includes the use of a log linear model, a method for training the model, and an algorithm for generating the top k candidates, whether there is or is not a predefined dictionary. The log linear model is defined as a conditional probability distribution of an output string and a rule set for the transformation conditioned on an input string. The learning method employs maximum likelihood estimation for parameter estimation. The string generation algorithm based on pruning is guaranteed to generate the optimal top-k candidates. This method is applied to correction of spelling errors in queries as well as reformulation of queries in web search. Experimental results on large scale data show that the proposed approach is very accurate and efficient improving upon existing methods in terms of accuracy and efficiency in different settings.

In our work,we are using effective phrase indexing to improve speed of performance to instant query search.

IMPLEMENTATION DETAILS

Limitations of Existing System

A main disadvantage of existing approach is that its perfor- mance can be low if there are many results matching the query keywords, which may take a lot of time to compute, rank, and sort. Thus it may not meet the high-performance requirement in an instant-search system.

To solve this problem, we propose a technique that can find the most relevant answers without generating all the candidate answers. In this approach, the inverted list of a keyword is ordered based on the relevancy of the keyword to the records on the list. This order guarantees that more relevant records for a keyword are processed earlier.

In order to support term proximity ranking in top-k query processing, introduces an additional term-pair index, which contains all the term pairs within a window size w in a document along with their proximity information.

Given a query q = ht1,t2i, if the index contains the pairs (t1,t2) or (t2,t1), their inverted lists are processed, their relevancy scores are computed based on the linear combination of content-based score and the proximity score, and the temporary top-k answer list is maintained. Then the top- k answer computation continues with the inverted lists of single keywords t1 and t2. Since the answers computed in the first step have high proximity scores, the early termination condition can be quickly satisfied in the second step.

Finally, a phrase is a sequence of keywords that has high probability to appear in the records and queries. We study how to utilize phrase matching efficiently to improve ranking in this top-k computation framework.

Proposed System Architectue

Fig. 1 shows proposed architecture of system indicating computation of valid phrases to the user query continuously when user interacts.These collected phrases are will be stored in dataset for further use of comparison when user enters keywords as a query.

We are preparing valid phrase based indexing and making tree based structure to store these phrases to fast retrieval when user enters a query.Next step is to develop effective query plan that helps to generate valid segmentation and ranking only top-k answers to the query.

Proposed system will overcome limitation of existing system as we are dealing with minimizing top-k answers by effective phrase indexing and segmenting those phrases in proper order.We are preparing an threshold value for the top-k answer that will help to ignore unwanted search of documents.

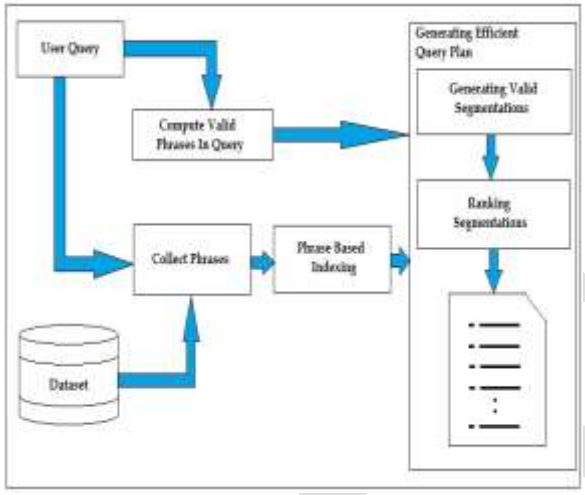

Fig.1 Proposed System Architecture

Fig.2 Mathematical Model for Proposed System

- **Ranking Segmentations with proper allocation:**

When we collect all generated segmentation for desired phrases, a way of accessing the indexes to compute its answer set. Then Query Plan Builder will rank these segmentations to decide the final query plan, which is supposed to be an order of segmentations to be executed. We can run these segmentations one by one until we find enough answers the query. Thus, the ranking needs to guarantee that the answers to a high-rank segmentation are more relevant than the answers to a low-rank segmentation. There are different methods to rank a segmentation.Segmentation comparator used to decide the final order of the segmentations. This comparator compares two segmentations at a time based on the following features and decides which segmentation has a higher ranking: Firstly,it will points summation between phrases and compares these phrases;Also,checks number of phrases available in the segmentation. The comparator ranks the segmentation that has the smaller minimum edit distance

summation higher. If two segmentations have the same total minimum edit distance, then it ranks the segmentation with higher value. As an example, for the query q = hhart,surgeryi, consider the segmentation “hart | surgery” with two valid phrases. Each of them has an exact match in the dictionary D, so its summation of minimum edit distances is 0. Consider another segmentation “hart surgery” with one valid phrase. This phrase has an edit distance 1 to the term “heart surgery”, which is minimum. Using this method, we would rank the first segmentation higher due to its small total edit distance. If two segmentations have the same total minimum edit distance, then we can rank the segmentation with fewer segments higher. When there are fewer phrases in a segmentation, the number of keywords in a phrase increases. Having more keywords in a phrase can result in better answers because more keywords appear next to each other in the answers.If two segmentations have both the same total minimum distance and the number of phrases, then we assume they have the same rank.

It is noticed that the answers to the segmentation where each keyword is a separate phrase include the answers to all the other segmentations. Therefore, once this segmentation is executed, there is no need to execute the rest of the segmen- tations in the plan. In the q = hhart,surgeryi example, the segmentation “hart surgery” is discarded from the query plan since the segmentation “hart | surgery” is ranked higher due to its smaller edit distance.

Basically,our approach to reducing time to rank answer documents using effective phrase indexing and proper segmentation.

- **Query Time for Proposed System:**

We can compare existing systems query time is much larger. Our proposed system works on indexing valid phrases and retrieving these valid phrases from the databae which is already stored.We are proposing segmentation of query using effective query plan.Proposed system is designed in a such a way that it take only specified threshold answer sets.

Fig.3 shows the relation between the number keywords and computational time in millieconds.The time required to retrieve ranked documents using instant fuzzy search by applying effective phrase indexing and segmenting those phrases with proximity ranking is minimum than existing sytem.

CONCLUSION

In this paper we study how to improve instant-fuzzy search by effective phrase index identification and segmenting those phrases with proper indexing by considering proximity information when we need to compute top-k answers. We studied how to adapt existing solutions to solve this problem, including computing valid phrases,computing all answers, doing early termination, and indexing term pairs. We proposed a technique to index important phrases to avoid the large space overhead of indexing all word grams by effective phrase identification and segmenting. We compared our techniques to the instant fuzzy adaptations of basic approaches. We conducted a very thorough analysis by considering space, time, and relevancy tradeoffs of these approaches. In particular, our experiments on real data will show the efficiency of the proposed technique for maximum of

2-keyword and for 3-keywords for some queries that are common in search applications. We concluded that minimizing top-k answer to the user query.

ACKNOWLEDGEMENT

I would like to extend my sincere thanks to all of them who encouraged and supported me for doing this work. I am highly indebted to Prof.Vinod S. Wadne,Ass. Professor, ICOER for his guidance and constant supervision as well as for providing necessary information regarding the project & also for his support.

REFERENCES:

[1] Ruihua Song, Liqian Yu, Ji-Rong Wen, and Hsiao-Wuen Hon "A Proximity Probabilistic Model for Information Retrieval", Microsoft Research Asia, Beijing, 100190, China.

[2] A. Nandi and H. V. Jagadish, "Effective phrase prediction," in *VLDB*,2007, pp. 219–230.

[3] M. Hadjieleftheriou and C. Li, "Efficient approximate search on string collections," *PVLDB*, vol. 2, no. 2, pp. 1660–1661, 2009.

[4] 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.

[5] Bhaskar Mitra, Milad Shokouhi, Filip Radlinski, Katja Hofmann," On User Interactions with Query Auto-Completion", Microsoft Cambridge, UK.

[6] A. Meenahkumary, V. Manjula, B. Divyabarathi, V. Nirmala," Top K Pruning Approach to String Transformation", 2014.

[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.

[8] Inci Cetindil, Jamshid Esmaelnezhad, Taewoo Kim and Chen Li," Efficient Instant-Fuzzy Search with Proximity Ranking", 2014.

[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] 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. 771–780.

[11] S. B¨uttcher, C. L. A. Clarke, and B. Lushman, "Term proximity scoringfor ad-hoc retrieval on very large text collections," in *SIGIR*, 2006, pp.621–622.

[12] H. Zaragoza, N. Craswell, M. J. Taylor, S. Saria, and S. E. Robertson,"Microsoft cambridge at trec 13: Web and hard tracks," in *TREC*, 2004.

[13] A. Arampatzis and J. Kamps, "A study of query length," in *SIGIR*,2008, pp. 811–812.

[14] D. R. Morrison, "Patricia - practical algorithm to retrieve information coded in alphanumeric," *J. ACM*, vol. 15, no. 4, pp. 514–534, 1968.

[15] 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.

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

[17] R. Schenkel, A. Broschart, S. won Hwang, M. Theobald, and G. Weikum, "Efficient text proximity search," in *SPIRE*, 2007, pp. 287– 299.

[18] 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.

[19] A. Jain and M. Pennacchiotti, "Open entity extraction from web search query logs," in *COLING*, 2010, pp. 510–518.

[20] Z. Bao, B. Kimelfeld, and Y. Li, "A graph approach to spelling correction in domain-centric search," in *ACL*, 2011.

[21] J. R. Herskovic, L. Y. Tanaka, W. R. Hersh, and E. V. Bernstam, "Research paper: A day in the life of pubmed: Analysis of a typical day's query log," *JAMIA*, vol. 14, no. 2, pp. 212–220, 2007.

[22] H. C. Ozmutlu and F. _Cavdur. Application of automatic topic identi_cation on excite web search engine data logs. Inf. Process. Manage., 41:1243,1262, September 2005.

[23] S. Ozmutlu. Automatic new topic identi_cation using multiple linear regression. Inf. Process. Manage., 42:934{950, July 200