International Journal of Engineering Research and General Science Volume 3, Issue 5, September-October, 2015
ISSN 2091-2730

Instant fuzzy search with proximity

Bhagyashri G. Patil, Sushilkumar N. Holambe

Perusing M.E. at Dr. B. A. M. U., Aurangabad

Perusing Ph.D. at Dr. B. A. M. U., Aurangabad

E-MAIL:BHAGYAPATIL11@GMAIL.COM, E-MAIL:SNHOLAMBE@YAHOO.COM

Abstract— In the present data warehousing environment schemes face lots and lots of issues to fetch out the resources with the help of referential or relational keyword terminologies. So the system requires some kind of advanced data manipulating schemes to extending the keyword search paradigm to relational data has been an active area of research within the database and information retrieval (IR) community. Instant search is important information retrieval technology for finding search results instantly as user query words. In instant fuzzy search it improves the searching results by comparing answers similar to typed query keywords. In this paper, we are using posstagger, soundex and wordnet for computing better results. All the existing systems contains basic solution is to compute all the answers and rank them but cannot meet the high speed requirements. All the existing systems contain basic solution for computing answers as user typing keywords, the previously searched results are not considered. So in this paper technique is proposed to use previously searched results.

Keywords- Auto complete, index, top-k, keyword search, edit distance, posstager, soundex

I. INTRODUCTION

As an emerging technology instant search returns answers to queries immediately based o user typed in. User many times makes search same queries. In this phase we cannot find correct answers as searched previously. In this case, by using fussy search technique user can find answers efficiently and correct [2]. Main computational challenge in this search technique is to achieve high speed requirement. As humans don't feel delay, from user typing queries character by character to the time result should be shown on device that full time is of 100 milliseconds. In instant search every query keystroke invokes question so it needs best speed to achieve high question turn over [3]. A Part-Of-Speech Tagger (POS Tagger) is a piece of software that reads text in some language and assigns parts of speech to each word (and other tokens), such as noun, verb, adjective, etc. Generally computational applications use more fine-grained POS tags like 'noun-plural'.

In auto completion system suggests many possible suggestions to user [4], [5]. Early termination techniques are used to minimize the number of possible results. Proximity ranking improve the results significantly [9], [10], [11], [6], [7], [8] .Our study shows to compute answers efficiently base on proximity of phrases.

Ranking

Here for this technique ranking is performed based on the relevancy of phrases into the records and frequencies and multiple co-occurrences of keyword in records. In [1] focus on the phrase matching in ranking operation. Basically three index structures are used inverted index forward index and tries [12]. Tries are used to show the keyword terms in dictionary D. In inverted leaf nodes are these dictionary terms. And forward index is for showing integers of each term of each record. These indexes are used for keyword matching a prefix condition.

Top-k answering

There are various methods to take top k answers first is computing all answers , which is useful for all types of ranking functions but it doesn't support when more matching keywords found . Second method is using early termination, which maintains heap for each keyword and it's monotonic. Third method is of using term pair in which is useful for only one or two keyword search.

II. PROPOSED WORK

A. Phrase indexing

To make early termination we must consider the records which match the query phrases, we have to access that records first. for example for query q=⟨xml data⟩, we have to access record containing the phrase "xml data" before the record containing "xml " and "data " separately. Index all available phrases based on mysql indexing strategy. In which B-tree indexing technique is used.

B. Finding valid phrases

In this section we compute the valid phrases based on post tagger technique. In which we are taking those phrases which consists noun and adjectives in typed queries. Most often user types or search name and previously cached valid phrases. Ontology based technique is used such as for finding semantic search soundx ad word net are used. Receiving a list of valid phrases, the Query Plan Builder computes the valid segmentations. The basic segmentation is the one where each keyword is treated as a phrase. Each generated segmentation corresponds to a way of accessing the indexes to compute its answers. The Query Plan Builder needs to rank these segmentations to decide the final query plan, which is an order of segmentations to be executed.

C. Top-k Query

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 segmentation. Our segmentation ranking relies on a segmentation comparator 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; 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 fewer segments higher.

D. Segmentation

Next step is to generate efficient segmentations, and that segmentation is nothing but phrase. User computed a query plan based on valid segmentations, and ran the segmentations one by one until top-k answers were computed. The database of segments and may be applied to an arbitrary text, preferably query, for splitting it into segments according to a segmentation procedure. The procedure matches all possible subsequences of the given tokenized query segmentation it is meant to segment the input query into segments, typically natural language phrases, so that the performance of relevance ranking in search is increased. For example query q=⟨"xml query processing"⟩, efficient segmentation is "xml | query| processing". If more phrases in the query then there more segmentations. In previous example "xml query" is valid phrase and "xml query | processing" is possible efficient segmentation. Many combinations of valid phrases are takes place if we consider more than three keywords or phrases from all valid segmentations the user typed keywords are searched. As shown in table 1, all possible efficient segmentations.

E. Ranking

These segmentations must be rank in order to get appropriate ranked results. Ranking is based upon the number of keywords in the segmentation and average nearest edit distance of valid phrases.

Table 1: three segmentations for query q=⟨"xml query processing"⟩.

1	"xml query processing"
2	"xml query |processing"
3	"xml | query | processing"

In this technique we calculate average distance of two or three keywords present in one record, according to those results are computed. At the time of ranking soundx is used for calculating semantic search of keywords. Word net library is used for semantic search which takes semantic keywords from database.

III. CONCLUSION

In this paper, we study, how efficiently integrate proximity information to ranking to compute relevant top –k answers. We use technique to find important phrases by avoiding considering large space overhead. And next is to compute and rank segmentations

including all indexed phrases. This technique is useful for 2-keywod or 3-keyword and more than 3-keyword search which is common. We concluded that computing all answers for all queries gives good performance and satisfy high speed requirement demand of instant search.

REFERENCES:

[1] Inci Cetindil, Iamshid Esmaelnezhad, Taewoo Kim, and Chen Li, "Efficient instant fuzzy search with proximity raking," in ICDE, 2014.
[2] Centennial, J. Esmaelnezhad, C. Li, and D. Newman, "Analysis of instant search query logs," In WebDB, 2012,pp.7-12.
[3] 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.
[4] K. Grabski and T. Scheffer, "Sentence completion," in SIGIR, 2004, pp.433-439.
[5] A. Nandi and H.V. Jagadish, "Effective phrase prediction," in VLDB, 2007, pp.219-230.
[6] R. Schenkel, A. Broschart, S. wonHwang, M. Theobald, and G. Weikum, "Efficient text proximity search," in SPIRE, 2007, pp. 287-299.
[7] H. Yan, S. Shi, F. Zhang, T. Suel, and R. Wen, "Efficient term proximity search with term pair indexes," in CIKM, 2010,pp. 1229-1238.
[8] M. Zhu, S. Shi, F. Zhang, T. Suel, and R. Wen, "Can phrase indexing helps to non-phrase queries?, "in CIKM, 2010, pp. 1229-1238.
[9] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algorithm for middleware," in PODS, 2001.
[10] F. Zhang, H. Yan, S. Shi, and R. Wen, "revisiting globally sorted indexes for efficient document retrieval," in WSDM, 2010, pp. 371-380.
[11] M. Persin, J. Zobel, R. Sacks-Davis, "Filtered document retrieval with frequency-sorted indexes," JASIS, val. 47, no. 10, pp. 749-764, 1996.
[12] S. Ji, G. Li, C. Li, and J. Feng, "efficient interactive fuzzy keyword search," in WWW, 2009, pp.371-380