International Journal of Computer Applications (0975 – 8887)
Volume 129 – No.5, November2015

Efficient Proximity Search with Query Logs

Sushilkumar N. Holambe
Perusing Ph.D. at DR. B. A. M. U.,
Aurangabad,
India

Bhagyashri G. Patil
Perusing M.E. at DR. B. A. M. U.,
Aurangabad
India.

ABSTRACT

In information retrieval technology there are various techniques for fetching data from resources. And that technique also contains various issues. Information retrieval techniques require advanced manipulating schemes which improves keyword search. There are many techniques have been proposed but results get down when large amount data interrupted. In this paper, have tendency to achieve efficient time and space complexities by integrating proximity information. This system improves the performance by using previous searching results. All the previous system consist basic solutions for extracting results and ranking them. Query logs consists the last searching results and use that results for next search. Fuzzy keyword search truly enhance the system usability. Existing system in databases requires to write complete keyword for searching but by using auto-complete scheme it is easy to type less and find more. In this system proper demand paging algorithm is used for finding previous results.

General Terms

Algorithm, Performance.

Keywords

Auto complete, Algorithm, demand paging, Top-k, Segmentation, Term pair, edit distance.

1. INTRODUCTION

Fuzzy search further improves user search experiences by finding relevant answers with keywords similar to query keywords. A main computational challenge in this paradigm is the high speed requirement, each query needs to be answered within milliseconds to achieve an instant response and a high query throughput, and it cannot meet this high-speed requirement on large data sets when there are too many answers[1],[2],[3],[4],[5].

The performance of the proposed techniques on real data sets. We implemented the following methods:

(1) Find All (“FA”): We found all the answers and returned the top-k answers after sorting them based on their relevancy score.

(2) Query Segmentation (“QS”): In this approach, we computed a query plan based on valid segmentations, and ran the segmentations one by one until top-k answers were computed.

(3) Term Pair (“TP”): In this approach, term pairs are used which increases as the window size increase.

2. FLOW OF WORK

2.1 Check user login

After User has login the personal details, user can check the details about their requirement, based upon their search result has produce within the milliseconds.

2.2 Valid phrases in a query

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 [6], [7]. 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. Here we are using log for improve search results. The previous search results must be part of next search history, so will put log/history in table. This can retrieve pages efficiently as query keyword requirement.

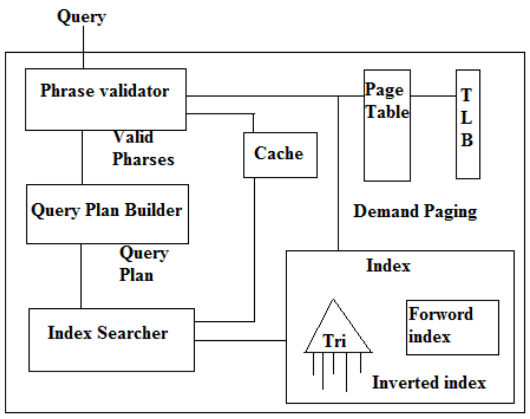


Fig1.Server architecture for Instant search

2.3 Top-k Query

The ranking needs to guarantee that the answers to high-rank segmentation are more relevant than the answers to low-rank segmentation. [8], [9] 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 [10], [11], [12].

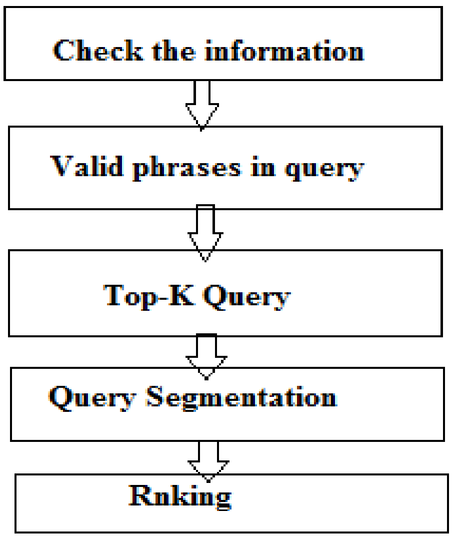


Figure 2 Flow of work

2.4 Query Segmentation

This paper compute 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 segments, typically natural language phrases , so that the performance of relevance ranking search is increased. Finally results are computed by using technique such as minimum edit distance.

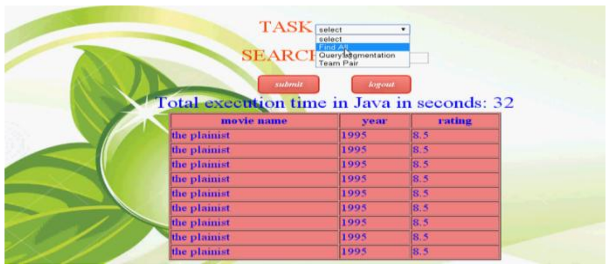
3. EXPERIMENTAL RESULTS

Experimental results consists time and task depending upon the previous systems used for instant search. The dataset used is movie datset .

3.1 Screen shots

1)Userlogin

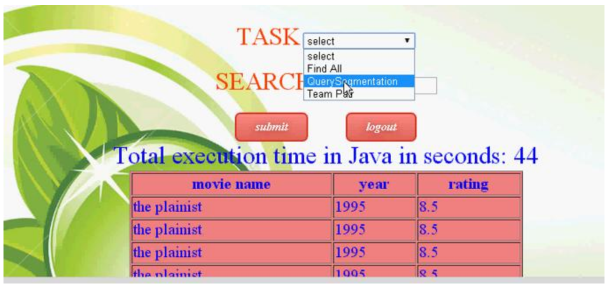
2) Find all



Total execution time in Java in seconds: 32

movie name	year	rating
the plainist	1995	8.5
the plainist	1995	8.5
the plainist	1995	8.5
the plainist	1995	8.5
the plainist	1995	8.5
the plainist	1995	8.5
the plainist	1995	8.5
the plainist	1995	8.5
the plainist	1995	8.5

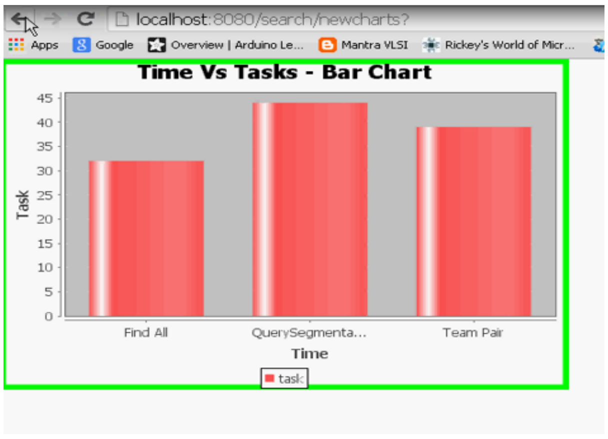
3) Query segmentation



Total execution time in Java in seconds: 44

movie name	year	rating
the plainist	1995	8.5
the plainist	1995	8.5
the plainist	1995	8.5
the plainist	1995	8.5

4) Final result based on graph



From experimental results conclusion is,

-Term pair is very slow.

-Find all is good for lengthy keyword search, but its slightly better for 1-keyword search.

- Query segmentation is good for 2-keyword or 3-keyword search as most of the applications use this and its better as the size of dataset increase.

4. CONCLUSION

In this paper, instant fuzzy search technique is compute relevant top-k answers. There are some techniques to find valid phrases by avoiding large space overhead. And next is computing valid segmentations which include all indexed phrases. In this paper page replacement strategy is used for using previously history / log. By using demand paging concept we can easily use log for previously searched results. Future scope consist semantic search on keyword may be joint keywords and searching more than 3 query keywords.

5. 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. won Wang, M. Theo bald, and G. Welkom, "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.