www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume – 5 Issue -02 February, 2016 Page No. 15784-15786

Efficient Instant Fuzzy Search With Proximity Ranking And Query Logs

Prof. Sushilkumar N. Holambe, Bhagyashri G. Patil

PG Coordinator ME(CSE), TPCT's College of Engineering,
Osmanabad, India
snholambe@yahoo.com

Perusing ME(CSE), TPCT's College of Engineering,
Osmanabad, India
bhagyapatil11@gmail.com

Abstract***: Instant fuzzy search is important developing technique from which users can find results character by character with better search experiences. The results must have high speed, good relevancy score and also good ranking functions used to get top results. Many functions are used to consider proximity of keywords which ultimately gives good relevancy score. In this paper, proximity information is used to ranking query results with which gives good time and space complexities. Many previously proposed techniques are used to achieve proximity ranking into instant fuzzy search. Most of the techniques firstly compute results and rank then according to some ranking functions, but if the dataset used is large then it takes time to compute all results and its very time consuming. At this state early termination technique is used to minimize space and time complexity. In this paper, incremental computation algorithm is used to overcome all drawbacks of previous systems and compute relevant results. Also query logs are used which are very useful for most of query suggestion systems, which ultimately reduces time complexity efficiently. The experimental results are computed to show space, time complexity and quality of results.***

Keywords: Auto complete, Algorithm, Term pair, edit distance.

1. INTRODUCTION

Instant search:

In Instant search as user typing query one or more possible matches are given to the user. Because of these immediate feedback users typing efforts minimizes. Suppose in IMDB movie dataset user typing query "Amit", the instant search system returns "Amit", "Amitabh", "Amita" and or "Amitabh Bachhan". Nearly all users want to reduce typing efforts [1]. There are two types of instant search systems as, instant-suggestion system and instant-result systems. Instant suggestion systems suggest the possibilities and user has to choose correct query and search. In instant result systems according to user's type query results are shown. As user's type incomplete query then result matching to these typed queries is shown. Example, if user types, "Hum" , the results maybe "Hum Tum" and "Hum Apake Hai kon".

Fuzzy search:

Fuzzy search is a technique in which if users make typographical mistakes and type wrong query spellings then automatically suggests correct queries and search that query. Exactly and highly matching matching results appear on top. Fuzzy search technique is like spelling corrector. Example, suppose user wants to type query as " Venkatesh" but by mistakenly user type as "wenkatesh" , then fuzzy search technique suggest correct spelling. Mostly on mobile phones typing mistakes are more because of small screen, at that time fuzzy search is important.

Related work:

Auto completion

In all user input systems auto completion is mostly useful technique. Auto complete technique not also used for one word but also for many words. There are many proposed methods for auto completion [2], Fuzzy tree structure and significant phrase prediction concepts are used.

Early termination

Many systems consists steps such as firstly find out all results and after that rank these results according to some ranking function. It is very insufficient to extract all matching results if the dataset used is very large. So early termination technique is used in which, sufficient results are extracted then searching stops. Early termination reduces time and space complexity [3], [4], [5].

Proximity ranking

Proximity ranking improves the relevancy and top results. Proximity ranking achieved by using different indexing method such as inverted index, forward index and tries. Proximity ranking can be achieved with early termination [3], [4], [6], [7].

In this paper main focus on incremental search with query logs. Query logs stores all the interactions of users with system. Different experimental results are conducted with this concept.

2. LITERATURE REVIEW:

2.1 Ranking

Most of the searching techniques use good ranking functions which are based on relevance score of query, query occurrences in datasets and distance between two query keywords in record. In proximity ranking main focus is on exact query matching as user type query.

2.2 Indexing

In instant search mainly three basic indexing techniques are used as inverted index, forward index and tries. Each query keyword is represented using tries. In tries each leaf node

presents the inverted list of prefix and in forward index records id is represented for each record according to inverted list [8].

2.3 Top-k answers

Most of the searching techniques prefer top-k result. Firstly results matching to query condition are computed and secondly these results are ranked based on their relevance score. The main disadvantage of this technique is results matching to query keywords are many which reduce the system performance by increasing time to extract query. So it does not give the higher performance. This disadvantage is removed by using early termination technique [9].

3. IMPLEMENTATION:

3.1 Server architecture of instant search:

In server architecture of instant fuzzy search, server receives requests and create inverted list of all query keywords present in dataset dictionary. Phrase validator mainly identifies the valid phrases present in dictionary. By comparing all matches and segmentations exact similar phrase is identified. Phrase validator computes valid phrases and based on trei structure inverted index is also computed. Query plan builder computes valid query plan which consists the valid segmentations with definite order. After query plan index searcher take each segmentation one by one and top-k results are computed. Cache module is used to store results computed by previous search and can be used to next query keyword search.

3.2 Computation of valid phrases:

As algorithm described in [10], incrementally valid phrases are computed. Algorithm explains how active nodes are computed from inverted list and from cache, computing valid phrase incrementally based on cached valid phrase of previous queries. In this, query logs are used before searching query incrementally. If user has previously searched the query, search logs are created and searching results are stored in log. Next time when user searching same query, then firstly searcher search for log if that query present in log then get retrieved if not then according to incremental search that query is searched. Query log reduces searching time for those queries which are previously searched.

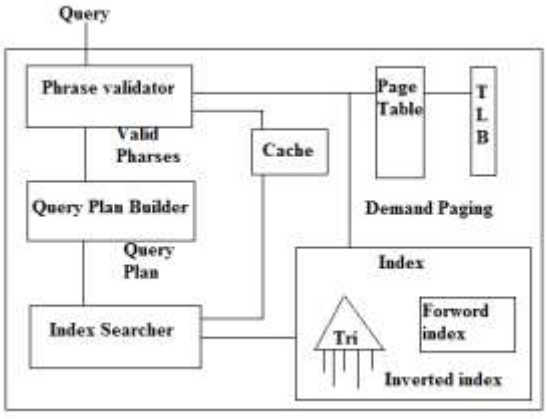

Fig.1: Server architecture of instant fuzzy search with query log

3.3 Segmenting and ranking queries:

From incremental computation valid phrase vector is computed. From valid phrase vector all possible segmentations are generated by using divide and conquer algorithm [10]. Example, suppose query q=< Life is beautiful > then possible segmentations are present in following table.

Table 1: Segmentations for query q.

1. "Life is beautiful"
2. "Life is | beautiful"
3. "Life | is | beautiful"

According to computed segmentations, these segmentations are ranked based on some ranking segmentations. Segmentations are ranked based on minimum edit distance and number of phrases present in segmentation. If any two segmentations have same minimum edit distance then they have same rank.

4. EXPERIMENTAL RESULTS

Experimental results consists time and task depending upon the previous systems used for instant search. For edit distance 1/3 as threshold value is used. Experimental results are conducted on windows 7, with 1.90 GHz RAM.

There are three datasets are examined IMDB movie dataset, IMDB book dataset and Enron dataset. Following methods are implemented:

1. Find all (FA)
2. Query segmentation (QS)
3. Term pair (TP)

Fig. 3: Efficiency of different approaches

Fig. 4: Final result based on graph

When no of datasets increases the computation time is also get increases. Cache hit for FA is better for all datasets. In 2-keyword search and 3-keyword search QA performs best, FA outperforms and TP mostly fail for 2 and more keyword search. 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.

5. Conclusion:

In this paper, incremental search with query logs is used. The previously proposed techniques are studied in this paper which consist incremental search, segmentation and proximity ranking. Query logs not only decrease execution time but also it helps users to find out appropriate results even if they have low partial knowledge. Query logs returns answers before typing full query and ultimately reduce typing efforts of users. Query logs are helpful for small devices as it reduce typing efforts.

References:

[1]. Centidil, J. Esmaelnezad, C. Li, and D. Newman, "Analysis of instant search query log," in WebDB, 2012, p.p.7-12.

[2]. Nandi, H. V. Jagadish, "Effective phrase prediction," in VLDB, 2007, p.p. 219-230.

[3]. R. Schenkel, A.Broschart, S.won Hwang, M. Theobald and G.Weikum, "Efficient text proximity search," in SPIRE, 2007,p.p. 287-299.

[4]. M. Zhu, S. Shi, N. Yu, J. R. Wen, "Can phrase indexing helps to predict non-phrase queries," in CIKM, 2008, p.p.679-688.

[5]. H. Yan, S. Shi, F. Zhang, t. Suel, and J. R. Wen, "Efficient term proximity search with term-proximity indexes," in CIKM, 2010.p.p. 1229-1238.

[6]. R. Song, M. J. Taylor, R. Wen, H. W. Hon, Y.Yu, "Viwing term proximity by different perspective," in ECIR, 2008, p.p. 346-357.

[7]. T. Tao, C. Zhai, "An exploration of proximity measure in information retrieval," in SIGIR, 2007,p.p. 295-302.

[8]. S. Ji, G. Li, C. Li, and J. Feng. "Efficient interactive fuzzy keyword search," in WWW,2009,p.p. 371-380.

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

[10]. Centidil, J. Esmaelnezad, C. Li, "Efficient instant fuzzy search with proximity ranking," in WebDB, 2014, p.p. 1-12.