International Journal of Engineering Research and General Science Volume 3, Issue 6, November-December, 2015
ISSN 2091-2730

An Instant-Fuzzy Search Using Phrase Indexing and Segmentation with Proximity Ranking

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— An Instant search is said to be effective when it gives faster retrieval of answer set with minimum computational time. Fuzzy search needed for queries that are mistypes due to several reasons. 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 and trie indices in the database. We study how to index these phrase threshold value and compare user threshold for effective answer set and develope 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,trie indices

Introduction

Finding relevant answers within time limit to a user query. Fuzzy logic is useful to retrieve exact documents to a user query while there may be misspelling in query. Instant search will helpful to lookup expected results while user typing a query.Searching to the target document is much easier and fast. Proximity ranking is responsible for getting relevant answer set to user query.Auto-completion of a query reduces time limit to retrieve relevant answer set to user query.Instant search 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. 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.

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.Text related interfaces have been undergoing a sea change in the last few years. An auto completion 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.

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]

Using trie index tree an improved fuzzy logic can be applicable to the words even though that mistypes. When user mistypes a query, even though system will retrieve most of the relevant answer set. Fuzzy searching is specially useful when researching unfamiliar, foreign-language, or sophisticated terms, the accurate spellings of which are not widely known. Fuzzy searching can also be used to locate individuals based on imperfect or partially inaccurate identifying information.

PROBLEM STATEMENT :

It is needed return proper results to an user query even user mistypes due to small interface just like mobile phones, lack of english spelling knowledge to user, etc. So, we study how to integrate proximity information into ranking in instant-fuzzy search to compute relevant answers to the query[8]. User mistypes a query due to small keyboards, mobile phones, spelling mistake or lack of subject knowledge, etc. 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]

Finding relevant answers within time limit to a user query. Fuzzy logic is useful to retrieve exact documents to a user query while there may be misspelling in query.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. Eventhogh user mistypes from at the start of query fuzzy search needed to calculate similarity as well as edit distance by considering trie index tree structure to get proper results.

LITERATURE SURVEY

In [1], In this paper, a document is altered to a pseudo document form, means containing long form, short forms or any related data to it. Also, a term count is propagated to other nearby terms, so that nearness among the words is find. Then they consider three heuristics, i.e., the distance of two query term occurrences, their order, and asigned term weights, which can be viewed as a pseudo term frequency.

In [2], author studied the problem of auto completion by considering different words. 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 auto completion 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 how to identify existing indexes, search algorithms, filtering strategies, selectivity estimation techniques and other work, and comment on their respective merits and limitations.

In [4], the author uses new early termination techniques for efficient query processing for the case where term proximity is integrated into the retrieval model. They implemented new index structures based on a term-pair index, and study new document retrieval strategies on the resulting indexes. They have performed a detailed experimental evaluation on new techniques and compare

them with the existing approaches. Experimental results on large scale data sets show that their techniques can significantly improve the efficiency of query processing.

In [5], author proved the results of lower ranked auto completion is basically receive lower engagement than the higher one. They suggested that users are most likely to engage with auto-completion after typing about half of the query, with particular at word boundaries. They also noticed that the auto-completion varies with the distance of query characters on the keyboard. Finally, the results indicates user engagement with auto completion is effective.

In [6], author implemented log linear model is indicating as 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 used to generate only important documents in answer set. Author also worked with error correction in query to provide effective answer set to the query.

In our work, we are using effective phrase indexing and also want the use of trie index structure to improve speed of performance to instant query search.

IMPLEMENTATION DETAILS

Proposed System Architecture

Fig. 1 shows proposed architecture of system indicating computation of dictionary and valid phrases to the user query continuously when user interacts. An trie indices are also generated for each of the valid phrases with an incremental approach.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.

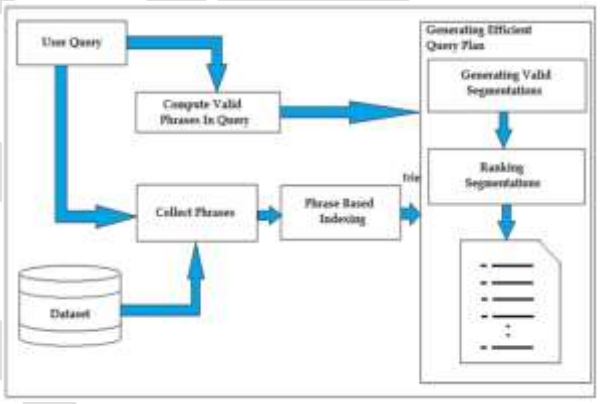

Fig.1 Proposed System Architecture

MATHEMATICAL MODEL

- **Finding Cosine Similarity Between Words :**

$$C.S. = \frac{\sum(Q_{wi}) * (P_{wi})}{\sqrt{\sum(Q_{wi})^2}. \sqrt{\sum(P_{wi})^2}}$$

Were,

C.S. = Cosine Similarity

Q = User query

W_i = Each word in the document

P = No. of occurrences of W_i in the document

- **Finding Edit Distance Between Keywords** :

e.g. "cats"

"ckats"

Edit Distance = 2

A DFA is mathematically represented as a 5-Tuple

(Q,∑, ∂, Q0, F)

The function ∂ is a transition function.

Fig.2 Shows mathematical model,

Were,

X1:User query

X2:Database

X3:Phrase Index Identification

X4:Collected Phrases

X5:Valid Phrases

X6:Effective Query Plan

X7: Gathering Valid Segmentation

X8:Ranking Segmentation

X9:Instan-Search Ranked Documents Choosen with Threshold Value

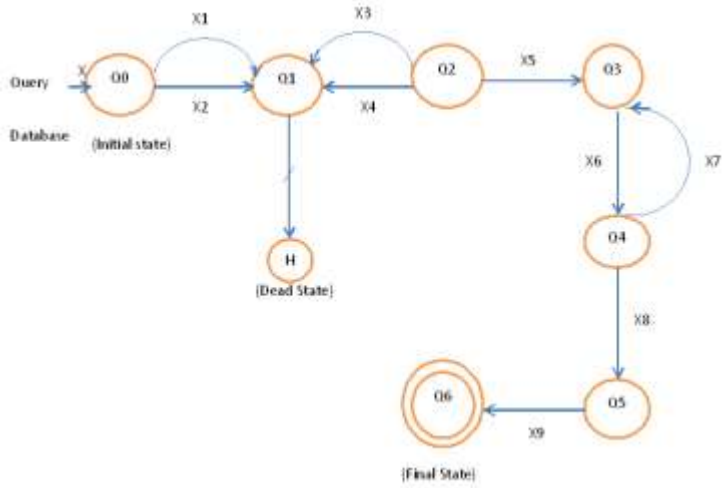

Fig.2 Mathematical Model for Proposed System

Modules

1. Module I: Loading Database to Instant Fuzzy Search

In this module, we are loading dataset of movies which is created in XML. XML document is loaded in to Instant Fuzzy Search for preparing overall dictionary to given dataset. Dataset is prepared from IMDB, were 2000 movies dataset is collected for real time processing.

2. Module II: Collection and preparation of Dictionary

After loading dataset, we are preparing dictionary which involves all the uniqe keywords from the XML document. Basically dictionary involves valid keywords after removal of stopwords. This prepared dictionary is passed to next phases of Instant fuzzy search.

3. Module III: Preparation Of Valid Phrases

Previous module stores unique keywords from dataset. On the basis of given dictionary, we prepare valid phrases. These phrases are important to retrieve relevant answer set to the user. These prepared valid phases are then forwarded to next phase for the preparation of Trie tree.

4. Module IV: Preparing Trie tree for Collected Valid Phrases

In this phase, we are preparing trie nodes which involves root and their various inserted childs depending on valid phrases. Basically, there is a possible chain of keywords which involves single parent node and multiple or single child to an active node. So, for the given valid phrases there are several active nodes involves several possible combination of childs.

Fig 1. Trie Index Tree

RESULTS

The IMDB dataset is used for the experimental setup that includes movies related information which is stored in structural format i.e. in XML format. We are using 2000 movies information from the IMDB dataset for our project. The given dataset is structured using separation of movie name, year of releasing and basic information.

TABLE I Movie Dataset

Dataset	IMDB(Movies Related Information)
No. of distinct keywords	5000
Average record length	40
Data Size	184 KB

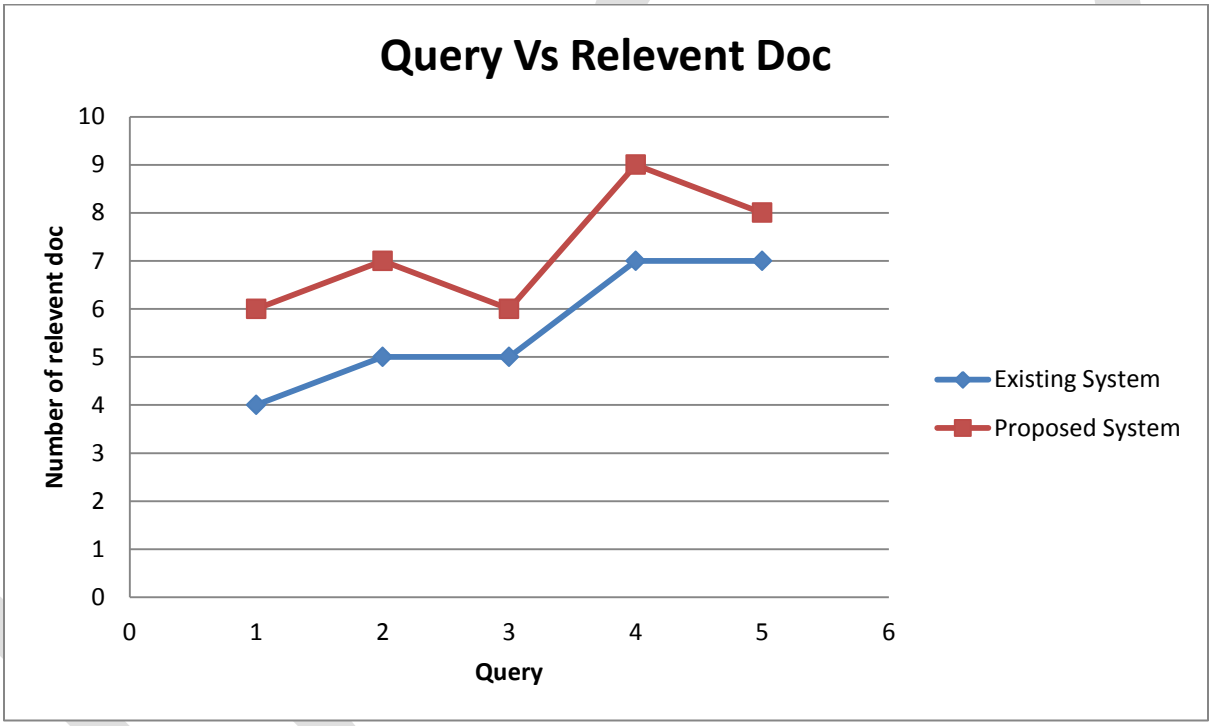

Fig. 3 Relevent documents to user query

The results obtained from the traditional Instant fuzzy search framework and implemented proposed system are taken on different sets of data. Both results are then compared to find out the conclusion. For both traditional and proposed system the same environment is used. The setup used for Instant fuzzy search and the results obtained are discussed below.

Figure 3 shows number for relevant document set is retrieved for given user queries. We are showing there are for five user queries as a input to the search engine and for each user query, search engine instantly gives responses. Figure 3 shows for the first user query, search engine gives us 6 relevant answer set from 10.(i.e. Total length of retrieved answer set to the user query). Similarly when user enters second query, that returns 7 relevant answer sets as shown in figure. Basically our proposed system will apply fuzzy logic from the first alphabet of the query as compared with existing system requires at least 3 characters.

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 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 dataset movies will show the efficiency of the proposed technique for retrieval of maximum no. of relevant document when user partially types a query. Even though user mistypes a query our proposed architecture retrieves maximum number of relevant answer set.

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 AutoCompletion", 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 2006