ISSN:2320-0790

An International Journal of Advanced Computer Technology

IMPROVED FUZZY SEARCHINGTECHNOLOGY

Mr. Shaikh Asharfali Ahmed[1], Prof. Ms. Megha Singh[2]

[1]M. Tech, Dept. of Computer Sci. and Engg., Central India Institute of Technology,Indore,MP, India

[2]HOD and Asst. Professor, Dept. of Computer Sci. and Engg, Central India Institute of Technology, Indore,MP, India

ABSTRACT: In web based applications information retrieval is gaining more popular. There are many advances of searching such as instant fuzzy search. Even if there are few typing errors in words the system will retrieve relevant data. Fuzzy search retrieves relevant data containing words which are similar to keywords. A main computational challenge in this paradigm is the high speed requirement, i.e., each query needs to be answered within milliseconds to achieve an instant response. Fuzzy search is implemented using Edit's Distance method.

Keywords: -fuzzy search, proximity ranking, edit distance, dictionary, inverted index, trie index

1. INTRODUCTION

User usually searches by providing the query. Queries contain a single or multiple keywords. IR i.e. Information retrieval is the method of obtaining necessary information from a collection of datasets. In this, search for a query will not show single result which match with the query instead of that it show many results will be shown. And the result set is very much important as the user will be interested in getting required information from this result set.

Users often make typing mistakes in their search queries. Meanwhile, small keyboards on mobile devicescan also cause mistakes. In this case we cannot find relevant information by finding records with keywords matching with the query exactly. This problem can be solved by supporting fuzzy technology, in which we find answers with keywords similar to the keywords.

To explain the importance of ranking let us consider the query "knowledge management". Systems that do not take ranking into account return general documents in which all the two terms knowledge and management are individually important, but the document does not necessarily contain information about knowledge management. On the other extreme an exact terms match would make sure that the document retrieved matches the query but implementing such phrase matching search would result in not displaying many relevant results. A proximity ranking, ranks query results based on distance between query keywords.

Edit distance method in combination with finds similar words faster. Edit distance refers to number of single character operations such as replacement, deletion or insertion need to be done in order to transform one word to another. For example edit distance between "George" and "Geordie" is two, since replacing character 'g' by 'di' word so the word "George" can be converted to "Geordie". Based on the length of the word a threshold for edit distance is determined all similar words within the threshold distance will be considered for fuzzy search.

2. PROBLEM STATEMENTS

Implementation of searching system with advanced search features such as fuzzy search technology with efficient searching. Existing search systems provide different kinds of ranking based on number of citations of the documents such as page ranking, ranking based on term frequency. Proximity is very important since it determine the relevant of the answers as search queries usually contain related keywords and user is mostly looking for data which have query keywords together.

3. RELATED WORKS

3.1 Fuzzy Search:

The studies on fuzzy search can be classified into two categories, trie-based approaches and gram-based approaches. In the former approach, sub-strings of the data are used for fuzzy string matching. This approach is especially suitable for instant and fuzzy search since each query is a prefix and trie can support incremental computation efficiently.The second class of approaches index the keywords as a trie, and rely on a traversal on the trie to find similar keywords.

3.2 Indexing

Inverted Index is most commonly used index; in this each word is mapped to a list of documents where the word is present. Indexing can play important role for faster retrieval of search results.Trie Index is a form of tree data structure, every node except leaf node consists of many branches each branch represents a particular character of the word. Forward Index is a type of index in which a document is mapped to list ofwords present in the document. As mentioned in Hyb indexing outperforms inverted indexing

by a factor of 15 – 20 in worst case. Hyb indexing is a variation of inverted lists in which data in compressed.

3.3 Proximity Ranking

In proposed a method to change the document score based on proximity of query terms. BM25 is a model of information retrieval which computes the score for document based on query terms by considering term frequency and inverse document frequency without considering proximity between query keywords. In proposed a method for proximity through spans. Depending on query terms spans are identified dynamically, the span need not be of fixed length and spans need not contain every query term. In the common phrases in the document are identified and they are stored as part of Trie data structure query is segmented into various segments using phrase information and documents are ranked based on phrase matching, one of the limitations with this approach is that the phrases need to be identified in advance as part of preprocessing step.

4. PROPOSED METHOD

4.1 Proposed system Architecture

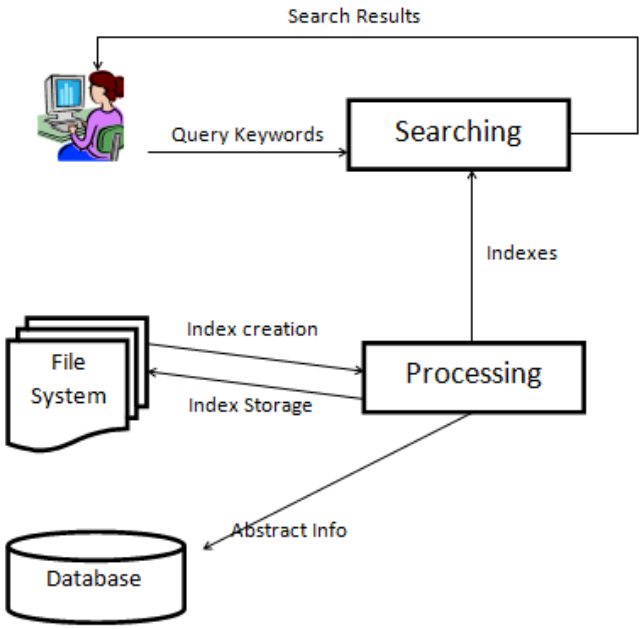

Fig. 1. System Architecture

Searching and ranking module is responsible for searching relevant result using keywords and indexes, this module ranks results based on proximity distance between query keywords. User uses the system to search relevant results. File system stores data set as well as index files.

Searching and Ranking

User can enter one query keyword or the query keyword can contain more than keyword for searching. When user enters more than one keyword proximity ranking will be enabled. Therefore fuzzy search is based on Edit distance.

$$D(i,0) = i, \quad \text{..... (1)}$$

$$D(0,j) = j, \quad \text{..... (2)}$$

$$D(i,j) = min \begin{cases} D(i-1,j)+1 \\ D(i,j-1)+1 \\ D(i,j-1) + \begin{cases} 1 & if\ x(i) \neq y(j) \\ 0 & if\ x(i) = y(j) \end{cases} \end{cases} \quad \text{.... (3)}$$

Fig. 2. Equation for Edit distance

Equation (1) and (2) represent base case equations for recursive function. In equation (3) the first term represent insertion cost of a character, 2nd term represent deletion cost of a character and third term represent replacement cost if two characters being compared are different.

Take the example of two strings, "George" and "Geordie"; how many characters would you have to change to transform "George" into "Geordie"? The answer is 2.

This is really only a minor tweak to the results to transform the number of edits required into a percentage.One modification of the results of the algorithm that I made to produce the goal I had was to produce a percentage instead of a number. To transform the results, I subtracted from 1.0 the number of changes required divided by the length of the longest string. Taking the resulting double value and multiplying by 100 yields a percentage; this step was only done to produce a result that would be meaningful to a user. Using the example of transforming "George" into "Geordie", the result would be: 1.0 – (2 / 7) or 71 (rounded down, remember the goal I had was to produce a fuzzy result and that 71 is close enough).

Before getting into the loop to calculate the edit distance, check to see if either string is empty. If one is, return the length of the other string. This points out one interesting note about Edit algorithm, in that the result is always bounded by the length of the longest of the two input parameter strings.

If neither string is empty, then construct the matrix for the calculation. The matrix should be the length of the first input string plus one by the length of the second input string plus one

		G	E	O	R	D	I	E
	0	1	2	3	4	5	6	7
G	1							
E	2							
O	3							
R	4							
G	5							
E	6							

Fig 3: Initial Matrix

Getting the loops run to completion results in a matrix that looks like:

		G	E	O	R	D	I	E
	0	1	2	3	4	5	6	7
G	1	0	1	2	3	4	5	6
E	2	1	0	1	2	3	4	5
O	3	2	1	0	1	2	3	4
R	4	3	2	1	0	1	2	3
G	5	4	3	2	1	1	2	2
E	6	5	4	3	2	2	2	**2**

Fig 4: The completed matrix

And the result is 2, as predicted originally.

Algorithm name-
FindSimilarWord-

orgDist[], editDist[] is array of length (word.length +1)
Initially orgDist has values 0,1,2,...

1. For each childNode of node
2. editDist [0] = orgDist[0] +1
3. for j =1 to word.length
4. editDist [j] = Apply Edit distance equation
5. childNode.orgDist = editDist
6. if(childNode. orgDist[word.length] >tr)
7. break
8. findSimilarWords(word, childNode, tr)
9. if(node.orgDist[word.length] <= tr)
10. add the wordId from Trie to similar word list.

Algorithm name-
Proximity ranking -
Steps:

1. For each list in wordIdList
2. for each docId in docIdList
3. obtain binId from inverted list
4. if(words are in same bin or words are in adjacent bin)
5. add the docId to the top of rankedList.
6. else
7. add the docId to the end of rankedList.

Above algorithm proximityRanking, is used to rank documents based on distance between query keywords. The entire set of documents which have all query keywords or words similar to query keywords are considered for ranking.

Algorithm Name: search
Steps:

1. remove stop words from query word list
2. apply stemming to each query keyword
3. for each keyword in queryWordList
4. find threshold edit distance
5. similarWordList = findSimilarWords(keyword, node, threshold)
6. documents with phrases = proximityRanking (similarWordList, wordIdList)
7. foreach keyword
8. find documents without phrases
9. Result = (documents with phrases) union (documents without phrases)

Above algorithm search, is used to search relevant documents. Initially preprocessing of query keywords is done, this involves removal of stop words from the keyword list. For each query keyword stemming is performed. To find list of similar words to each query keyword threshold distance is calculated based on length of query keyword. Similar words are found using Edit distance. Inverted lists are intersected to determine documents which contain all query keywords. Proximity ranking is applied to find documents with phrases. Documents without phrases i.e. Documents containing few query keywords but they do not form a phrase will be identified. The result will be displayed as union of documents with phrases and documents without phrases.

5. CONCLUSION

This system is developed to provide advanced search features like fuzzy search and proximity ranking. Fuzzy search helps user in retrieving relevant results even if there are few typographical errors in the query keywords. Proximity ranking ranks records based on distance between query keywords. Edit distance is used in fuzzy search. Proximity ranking makes use of modified inverted list by storing word positions in the form of bin Id. There are other advanced search features like auto completion, page ranking etc. which will be considered in future for implementation.

6. REFERENCES

[1]. InciCetindil, J. Esmaelnezhad, T. Kim and Chen Li "Efficient Instant-Fuzzy Search with Proximity Ranking„" IEEE 30 International conference on data engineering year 2014.

[2]. M. Persin, J. Zobel, and R. Sacks-Davis, "Filtered document retrieval with frequency-sorted indexes," JASIS, vol. 47, no. 10, pp. 749–764, 1996

[3]. A. Singhal. "Modern information retrieval: A brief overview." Bulletin of the IEEE Computer Society Technical Committee on Data Engineering.