APPROXIMATE PATTERN MATCHING WITH RULE BASED AHO-CORASICK INDEX

M.Priya[1], Dr.R.Kalpana[2]
[1]Professor, Department of Computer Science and Engineering
Bharathiyar College of Engineering and Technology, Karaikal
[2]Professor, Department of Computer Science and Engineering
Pondicherry Engineering College, Puducherry

ABSTRACT

Language processing application areas such as Information Retrieval, Information Extraction, and Machine translation etc are using pattern matching as the basic requirement for extraction of information from the documents. Nowadays, Searching through the multimedia database for the particular pattern is the biggest issues due to the tremendous amount of records to be dealt to search. On the internet, the pattern matching mechanism is used to retrieve the documents that are similar to the given query. The exact pattern matching mechanisms are not suitable for this situation since it searches only for perfect matches. So that the approximate pattern matching mechanism is introduced. The reduction in database search space is attainable by using appropriate data structure used for searching. Moreover, the entire pattern matching algorithms support the error detection and correction of the pattern to be searched, not in the searching text. In this paper, a new mechanism is proposed to search a single and multiple patterns in the given database and performs correction in the text if it contains misspelled word.

Keywords: Rule index, Context-dependent error, Trie, Dictionary based search, Error Correction

I. INTRODUCTION

The Internet has become an important component for billions of people for doing their business in both international and national level and additionally it is the most primary available worldwide information knowledge base to access the information. A search engine helps to navigate through the web pages to gather the information from the internet. The search engine does not search the whole internet however through the internet index. Internet index is formed by a tiny simple program called web crawler. Web crawler searches through all websites by following the links of the web pages and send the key information regarding the page to store it within the index. The key information could also be title, subtitle, Links and Meta tag etc. The web crawler cannot enter into certain websites because it is encoded with computer code. This kind of web is named as deep web.

Search engines use a search algorithm to retrieve information stored within the internet index. The appropriate search algorithm often depends on the data structure being searched, but also on any a priori knowledge about the data. The searching algorithm can be categorized into static and dynamic searching. Static search is used to retrieve the information from fixed records. Dynamic search allows retrieving the record from the databases which allow dynamic insertion, deletion of records. Nowadays, Searching through the database for the particular pattern is the immense issues due to the huge amount of records to be dealt to search. When the user types the query keyword on the web browser, the search engines search through the multimedia databases. The search engine retrieves a collection of records from the multimedia databases that match with the given query keyword. The search engine doesn't search through the whole databases but search happens only to the intended entry.

The search engines use pattern matching algorithm for performing searching operations. The Pattern matching is the act of checking and finding the location of the given sequence of token called pattern, in some text. The dictionary-based pattern matching algorithm gets importance in multimedia databases. Given a dictionary of words that are particular field selected as specified in the given query is called intended entry and a search key, the search happens through that particular entry. There is a possibility of having a real-time error or context-dependent error during creation of a database. In this paper, a survey is made about the searching algorithm and a new searching mechanism is introduced to search the given query keyword and also carry out for correcting the error if the database contains context dependent errors.

II. RELATED WORK

Context-dependent errors occurred because of human can be classified into typographical error or cognitive error. A typographical error is related to the keyboard error that happens because of typing misspelled the word. Cognitive error occurs due to the writers substitute to the wrong spelling of a homophone or near-homophone. These two types of error are called real-time errors and detecting such errors is called real-time error detection. The task of removing these types of errors is called context-sensitive error correction. Pattern matching is the basic mechanisms used in error detection and correction mechanism. Given two sequences T and P, the string search problem consists of determining all location of P occurs in T. The exact matching algorithms provide a perfect match. The exact matching algorithm doesn't provide a perfect match to retrieve the documents from the internet since the internet is the pool of information in terms of terabyte and this information are collected from heterogeneous information.

An approximate string matching algorithm is used to search a pattern on the internet. A survey [4] is made to study the different approximation algorithm and its statistical behavior is discussed. A standard technique for approximate string matching is edit distance [1,2] also known as k difference problems. The edit distance dist (P, T) is defined as the minimum cost requires transforming P into T. The minimum cost operations are an insertion, deletion, and substitution. For each operation, the cost or weight is assigned. The operation list or alignment is selected based on their minimum weight for finding the minimum string edit distance. This technique is applied as the basic technique for searching a pattern in various applications[5] like comparing two ink sequences using ScriptSearch algorithm, document retrieval based on hand written script image and so many image retrieval based on the given string etc.

The N-gram approach [6] to spelling error detection and correction in context sensitive applications is proposed by Mays et al. For given sequence S= {s_1, s_2, s_3, s_4… s_n}, replace any si by si', the alternative spelling. The alternative spelling sequence for correction is choosing in such a way that it maximizes the probability of correcting the spelling. The conditional probability with Baye's theorem is used to select the best alternative for the correction of errors. The N-gram model is used as a basic mechanism for so many applications that uses language processing such as text categorization, classification of text or clustering of text.

The performance of the searching techniques depends on the data structure used for searching, the search space pruning supported by the data structure. Most of the existing algorithm performs well in the detection of error but they are not supports for correction error. Some of the algorithms carry out the correction of the error but not in the expected efficiency. This paper discusses different tree based searching and also discusses how far it supports the concept of error correction.

III. APPROXIMATE PATERN MATCHING

The problem of approximate pattern matching is typically broken up into two sub-problems one is finding approximate substring matches inside a given string and the second one is finding dictionary strings that match the pattern approximately. The most basic approach called naïve algorithm for intended entry is to loop through the text phrases in the entry, and searches for the pattern in each phrase, one by one. This approach does not scale well. Searching for a string inside another set of strings has the complexity of $O(n)$. Repeating that for m search phrases give the time complexity of $\boldsymbol{O(m * n)}$.In this approach, text phrases in the entire dictionary have to be searched for the given pattern. The tree-based index structure is best for

string searching since it is possible to prune the maximum search space by the cut off the other path except for the search path.

A. Trie

A Trie has also known as a prefix tree, is a special type of tree used to store associative data structures. Trie or prefix based tree is a tree data structure for searching multiple patterns of strings. It acts as a fundamental block in string processing algorithms. Each node represents a character or a word. If two strings have a common prefix, then the trie has the common ancestor for them. It is an ideal data structure for storing the information in lexicographical order. The phrases are indexed in a structure that can be traversed in a linear manner with the given pattern in parallel and completes the search within a single pass.

The construction time of the trie is ***O(lx)*** where ***l*** is the average length of the prefix of the text and ***x*** is the total number of words used. The search complexity of trie for the given pattern is **O (n)** where n is the length of the given pattern. It has the benefits of the hash table in terms of the space complexity. Trie offers excellent performance in space and searches time, only when the databases consists of strings of the common ancestor. Orthographic corrector program in search engines uses the prefix based trie for providing the possibilities of correction for the misspelled word. It lists all the prefix of the word in the web browser. It has the disadvantages of applying the correction only for the given pattern, not for the words in the databases

B. An Aho_Corasick Matching algorithm

An aho_corasick string matching algorithm is a powerful string matching algorithm proposed the best complexity for multiple patterns. It is a special form of trie. It is an optimal searching algorithm for both single and multiple pattern searching. Suppose the intended database entry consists of M phrases of a maximum length of text T, and the given N pattern of search with length $L=\{L_1,L_2,L_{3,}\text{---}\ L_N\}$.The direct matching algorithm supports the concept of searching the first pattern through the M phrases of the database entry, then search the second pattern. Repeat this procedure form N pattern. It leads to the time complexity of ***O(N*M+L+T).***

In general, the time complexity of the aho_corasick algorithm for a single phrase is ***O(N+L+Z)*** where Z is the count of output matches. Various data structures are used to implement this algorithm such as array, tree data structure and Hash table. Each one is having its own disadvantages of requiring extra memory. For instance, if the algorithm is implemented using the array, the additional space of *O (L*q)* is required, and the time complexity will be *O((N+L)*q +M)* where q is the length of the alphabet. If the tree based data structure is used, then the additional memory required is *O(log(q))* and the time complexity is *O((N+L)*log(q)+Z).* If the same algorithm is repeated for a text of M phrases, then complexity will be *O(M*(N+L+Z)).*

To reduce the space and time complexity, an aho_corasick algorithm based on trie is implemented to search a pattern or multiple patterns within the dictionary of words that is called as the intended database entry. Moreover, an aho_corasick tree algorithm for the exact matching problem.

IV. PATTERN MATCHING USING AN AHO_CORASICK

This model is used to perform web searching in order to retrieve the document by an approximate keyword search mechanisms. Up to the previous survey, there is an algorithm to correct the search pattern if it has an error, but there is no algorithm that supports the misspelling of words in the searching text. Consider the model consists of words from the intended database $W=\{W_1, W_2, W_3\text{—}W_M\}$and the pattern to be searched will be P. The system architecture is shown in Figure 1

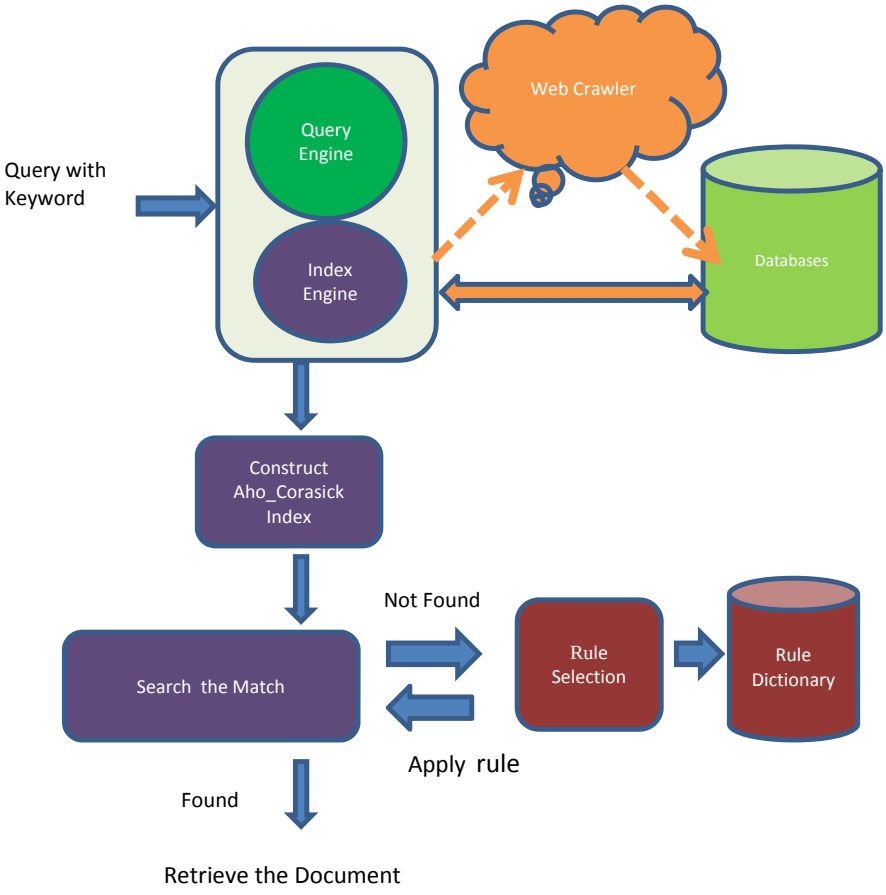


Figure1.Architecture diagram for String Matching

The user enters the query keyword within the web browser. The web crawler is a program that searches through the web pages and extracts the information. The index is constructed using the aho_corasick tree. The given query keyword is search through the tree from the root. If the match is found, then retrieve the corresponding documents. If the match is not found, then refer the rule dictionary, find the appropriate rule and replace the node. If the new character matches with the character in the keyword, then the corresponding document is retrieved. Repeat this procedure until the match found. If the current path is failed, then the failed path within the tree is used to try other alternatives. In order to support for multiple patterns, there is a possibility of dividing the tree into so many chunks for parallel processing.

The Figure 2 a shows some of the words from the hotel name and Figure 2 c shows the corresponding aho_corasick tree.

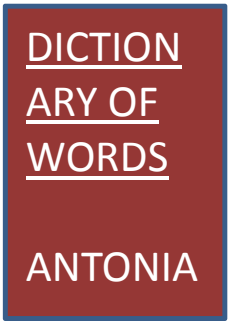
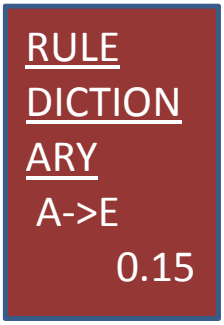


Figure 2 a. Key Field



2b.Rule dictionary

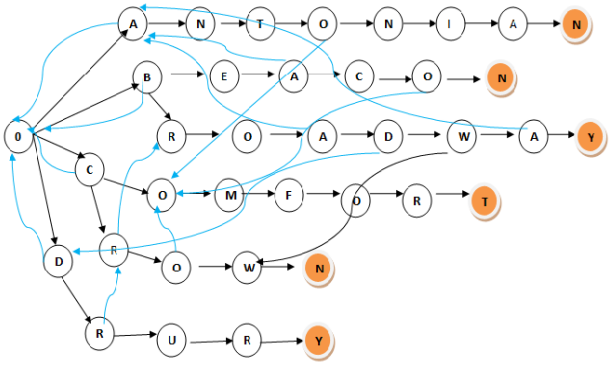


Figure 2c. An Aho_Corasick tree

Figure 2b shows the context-based rule dictionary which is created with the help of training data set during offline. The rule dictionary consists of rules and each rule is associated with the corresponding weight. The weight of each rule is calculated by probabilistic based logistic or log-based model. Suppose the given pattern is "BAACON". The search starts from the root O. Match with the node B; go to next node. E is not equal to A. Then refer rule dictionary. Rule E->A is found. Replace the node with E with A and continue the matching. If no match found in the rule dictionary, then follow the failure link. The time complexity of the algorithm will take O(R) extra time for rule selection and additional space for storing the rule dictionary.

V. EXPERIMENTAL EVALUATION

The keywords are collected from the Hotel dataset jeon. The offering.zip.txt file consists of the database information like hotel name, phone number, address etc. From the dataset name of the hotel, field is retrieved for searching. The efficiency of the pattern matching algorithm is affected by two factors 1) Size of the rule dictionary 2) Response time

The number of matches to be found for the query word depends on the size of the rule dictionary. The number of matches increases with the size of the rule dictionary as shown in Figure 3.So that the time to find the match also increases.

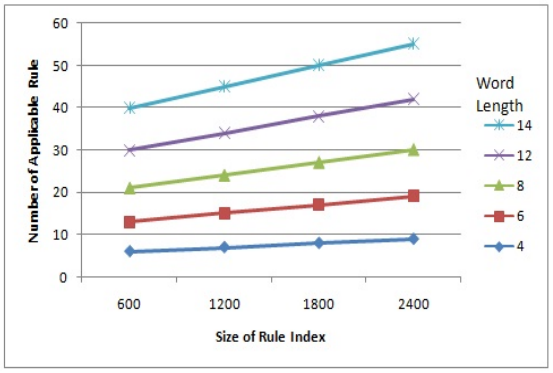


Figure 3.Number of Applicable Rule Vs Size of the Rule Index

The time complexity of the algorithm depends on the effectiveness of the aho_corasick tree. The time complexity of the aho_corasick tree can be categorized into tree construction time and the searching time. Tree construction time is ***O(ln)*** where ***l*** is the length of the longest word in the intended entry of the database and ***n*** is the number of records. The searching time of the algorithm depends on the length of the query and the number of matches found in the rule dictionary and the rule dictionary size. The response time increases with the size of rule dictionary as shown in Figure 4

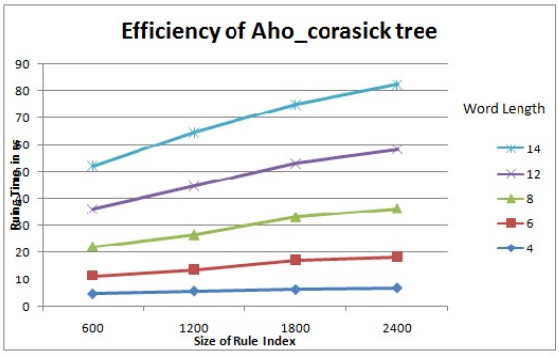


Figure 4.Efficiency of Pattern Matching

Not only the efficiency, but also the accuracy of the system depends on the selection of the best rule applicable for the correction. Otherwise there is a possibility of improving false positive and negatives.

VI. CONCLUSION

The aho_corasick tree-based index structure is constructed for searching a single and multiple patterns within the given database. The efficiency of the searching is discussed with the different data structure. The index structure has the advantage over others in terms of pruning technology used and the time complexity. It has a drawback of space complexity, which requires extra space to store all the rules related to context. In future, this type of searching technique can be associated with any other index structure to improve the efficiency in searching the internet.

References

[1] Carl Barton, Costas S Iliopoulos and Solon P Pissis, "Fast algorithms for approximate circular string matching", published in Algorithms for Molecular Biology 2014**9**:9 https://doi.org/10.1186/1748-7188-9-9
© Barton et al.; licensee BioMed Central Ltd. 2014

[2] Cong, G. Jensen, C.S and Wu, D. "Efficient retrieval of the top-k most relevant spatial web objects," Proc. VLDB Endow., vol. 2, pp. 337–348, 2009.

[3] De Felipe, I. Hristidis,V. and Rishe, N . "Keyword search on spatial databases," in Proc. Int'l Conf. Data Eng. (ICDE), 2008, pp. 656–665

[4] Gao, M. Jin, C. Wang, W. Lin, X. and Zhou,A." Similarity query processing for probabilistic sets", in International Journal of Information Sciences, 196:97–117, 2012

[5] Ge,T. and Li,Z. " Approximate substring matching over uncertain strings" in Proceedings of the VLDB Endowment, Vol. 4, No. 11, pages 772–782. VLDB Endowment, 2011.

[6] Gonzalo Navarro,"A guided tour to approximate string matching " in ACM computing surveys ,Volume 33 ,Issue 1 March 2001.

[7] Li, Z. Lee, K. Zheng, B. Lee, W,C. Lee, D.L. and Wang,X. "IR-Tree: an efficient index for geographic document search," IEEE Trans. Knowledge and Data Eng., vol. 23, no. 4, pp. 585–599, 2011.

[8] Lopresti, D. and Tomkins, A. "Block Edit Models for Approximate String Matching", in proceedings of the 2nd Annual South American Workshop on String Processing, pp. 11-26

[9] Nguyen Le Dang, Dac-Nhuong Le, and Vinh Trong Le," A New Multiple-Pattern Matching Algorithm for the Network Intrusion Detection System", IACSIT International Journal of Engineering and Technology, Vol. 8, No. 2, April 2016.

[10] Suen, Ching Y., "N-Gram Statistics for Natural Language Understanding and Text Processing," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI- 1, No. 2, April 1979, pp.164-172.

[11] Wagner, R. A. " On the complexity of the extended string-to-string correction problem", in Proceedings of the 7th ACM Symposium on Theory of Computing, Association for Computing Machinery, 1975.

[12] Wagner, R.A. and Fischer, M.J. "The string-to-string correction problem", in journal of the Association for Computing Machinery, 21:168-173, 1974.

[13] Ziqi Wang, GuXu, Hang Li, and Ming Zhang "A Probabilistic Approach to String Transformation ", IEEE transaction on knowledge and data engineering, 2013