TEXT ANALYZER

G. Arumugam [#], M.Thangaraj [$], R. Sasirekha*

[#]*Prof. & Head, Department of Computer Science*
Madurai Kamaraj University, Madurai – 21.
[1]gurusamyarumugam@gmail.com

[$]*Assosiate prof. Department of Computer Science*
Madurai Kamaraj University, Madurai – 21.
[2]thangarajmku@yahoo.com

**Research Assistant, SSE Project, Department of Computer Science, Madurai Kamaraj University*
Madurai – 21.
[3]sasirekhars@gmail.com

ABSTRACT

The web has become a resourceful tool for almost all domains today. Search engines prominently use inverted indexing technique to locate the web pages having the users query. The performance of inverted index fundamentally depends upon the searching of keyword in the list maintained by search engine. Text matching is done with the help of string matching algorithm. It is important to any string matching algorithm to locate quickly the occurrences of the user specified pattern in large text. In this paper a new string matching algorithm for keyword searching is proposed. The proposed algorithm relies on new technique based on pattern length and FML (First-Middle-Last) character match. This proposed algorithm is analysed and implemented. The extensive testing and comparisons are done with Boyer-Moore, Naïve, Improved Naïve, Horspool and Zhu Takaoka. The result shows that the proposed algorithm takes less time than other existing algorithm.

KEYWORDS

Web searching, String matching, Syntactic IR

1. INTRODUCTION

Internet is potentially the world's largest knowledge base and finding information in the large knowledge base is very difficult by surfing internet information globe. Search Engines emerged and developed quickly in this background. Search engine is a very important tool for people to obtain information on internet. Day by day the quantity of information is increasing exponentially on the internet. According to survey of net cafe, the web has crossed 110 million sites in March 2007[6] [8] and as of November 2009 there are about 20,340,000,000 WebPages are crawled [11].With this exponentially growing information size on the Internet, utilization of information has become major focus. And so search engine developers began to pay attention to the quality and relativity of searching results. There are three types of search engines whose searching technique is different [10].

1.1 Contents Search Engines

Information mostly faces to websites, offering contents browsing and direct searching services, and users would query information limited to some contents. Because this kind of Search Engines join intelligence of people, information which was searched is accurate, navigation quality is good, and the defect is that they need to manually intervene, information is incomplete, and information updating is out of time.

1.2 Robot Search Engines

These search engines provide full text querying services. Robot program which is called Spider or Robot automatically collects information with width or depth strategy or other strategies on Internet, and collected information is stored in database, created index by indexer. Querying machine searches indexed database based on user's request, returning relatively queried results to users. The merits of this kind of Search Engines are that they don't need to manually intrude, vast information, and information updating is in time, while the defect is that returned results are excessive, having vastly irrelevant results, so users must choose relevant information from results.

1.3 Meta Search Engines

They don't analyze through Internet, not having themselves data, but submitting users queried requests to many Search Engines at the same time, after uniting, removing repetition, and renewing rank so unifying disposal, they return searched results to users. The merits of this kind of Search Engines are that they can provide comparatively general and correct information in short time, and defects are that not completely using functions of Search Engines, users need filter more. Search engine navigates efficiently through the information on the internet, offering the retrieval services for the users. Usually working of search engine can be divided into information gathering, indexing, query service and ranking. The performance of indexing module plays a major role in its overall performance. The conventional keyword searching algorithm has been used in response timing requirements of recent search engine [6] [8]. So research on this topic has been done which results in a fast retrieval keyword searching algorithm. Searching algorithm helps to retrieve information from pool of websites available in the internet. The two major approaches to Information Retrieval (IR) are syntactic IR and semantic IR.

Search terms are represented as arbitrary sequences of characters and information retrieval is performed through the computation of string similarity. The search procedure used by these search engines is principally based on the syntactic matching of document and query representations. These search engines are known to suffer in general from low precision [4]. In text retrieval, full text search refers to a technique for searching a computer-stored document or database. In a full text search, the search engine examines all of the words in every stored document as it tries to match search words supplied by the user. Full-text searching techniques became common in online bibliographic databases Most Web sites and application programs provide full text search capabilities. Some Web search engines, such as AltaVista employ full text search techniques, while others index only a portion of the Web pages examined by its indexing system. Text search functionality takes a set of keywords and locates them in a document or a database containing words. Search strings provided by users are usually context based. However, semantic technologies represent meaning using ontologies and provide reasoning through the relationships, rules, logic, and conditions represented in those ontologies.

2. GLOBAL SENARIO

A web search engine is a tool designed to search for information on the World Wide Web. The search results are usually presented in a list and are commonly called hits. A search engine operates, in the following order

Web crawling, Indexing and Searching

When a user enters a query into a search engine the engine examines its index and provides a listing of best-matched web pages according to some criteria, usually with a short summary containing the document's title and sometimes parts of the text.

2.1. Web Crawling

Web search engines work by storing information about many web pages, which they retrieve from the WWW. These pages are retrieved by a Web crawler — an automated Web browser which follows every link it sees [3] [7]. The contents of each page are then analysed to determine how it should be indexed (for example, words are extracted from the titles, headings, or special fields called Meta tags). Meta data about web pages are stored in an index database for use in later queries.

2.2. Indexing

Some search engines, such as Google, store all or part of the source page as well as information about the web pages whereas others such as AltaVista store every word of every page they found[3][7]. This cached page always holds the actual search text since it is the one that was actually indexed, so it can be very useful when the content of the current page has been updated and the search terms are no longer in it. Increased search relevance makes these cached pages very useful even beyond the fact that they may contain data that may no longer be available elsewhere.

2.3. Searching

When a user enters a query into a search engine the engine examines its index and provides a listing of best-matching web pages according to its criteria usually with a short summary containing the document's title and sometimes parts of the text. Most search engines support the use of Boolean operators AND, OR and NOT to further specify the search query. Some search engines provide an advanced feature called proximity search which allows users to define the distance between keywords.The usefulness of a search engine depends on the relevance of the result set it gives back. While there may be millions of WebPages that include a particular word or phrase some pages may be more relevant, popular, or authoritative than others [1]. Most search engines employ methods to rank the results to provide the best results first. Search engine decision to have the best matching pages and order of results vary widely from one engine to another. The methods also change over time as Internet usage changes and new techniques evolve.

2.4. Searching Technique

At the frontal end, browser is connected with the web server when users put forward a searching request to the browser. Web server can find the matched documents in a large-scale indexed database, lists the indexes of these documents, and returns the result to the user. At the hinder end of the search engine, administrator extracts the web page's

indexed information saves them with the web page's URL in the indexed database. The indexed database must be updated continuously to satisfy the dynamic changes of the Internet [10].The work flow of searching mechanism is shown in the diagram [Fig:2.4.1] below.

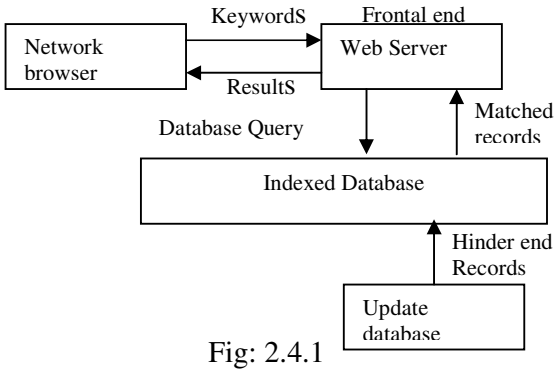

Fig: 2.4.1

3. RELATED WORK

3.1 String Matching Algorithm

The string matching algorithm is used to find the occurrences of a pattern P of length m in a large text T. Many techniques and algorithms have been designed to solve this problem. [2] These algorithms have many applications including Information retrieval, database query, and DNA and protein sequence analysis. Therefore the efficiency of string matching has great impact. Basically a string matching algorithm uses the pattern to scan the text. The size of the text taken from file is equal to length of pattern. It aligns the left ends of the pattern and text. Then it checks if the pattern occurs in the text and shifts the pattern to right. It repeats the same procedure again and again till the right end of the pattern goes to right end of the text. Two types of matching are currently in use. Exact string matching technique [12] [15] used to find all occurrence of pattern in the text. Approximate string matching is the technique of finding approximate matches of a pattern in a text. The closeness of the match is measured [4] [9].

The most obvious approach to string matching problem is Brute-Force algorithm, which is also called "naïve algorithm". This algorithm is simple and easy to implement, works good mostly for binary string search. It compares the pattern P with text T if mismatch occurs the pattern is shifted to one position right and again checks with the text T. This algorithm performs too many comparisons over the same text; hence it took more searching time. This algorithm is unacceptably slow

The Rabin-Karp string-searching algorithm calculates a hash value for the pattern, and for each M-character subsequence of text to be compared. If the hash values are unequal, the algorithm will calculate the hash value for next M-character sequence. If the hash values are equal, the algorithm will do brute- force comparison between M-character sequences. In this way, there is only one comparison per text subsequence, and character matching is only needed when hash values match. There are so many different strings, to keep the hash values small we have to assign some strings the same number. This means that if the hash values match, the strings might not match; we have to verify that they do, which can take a long time for long sub strings

Knuth, Morris and Pratt [13] discovered first linear time string-matching algorithm by following a tight analysis of the naïve algorithm. Knuth-Morris-Pratt algorithm keeps the information that

naïve approach wasted gathered during the scan of the text. By avoiding this waste of information this turns out to be the length of the longest *prefix* of fewer than *x* characters in the pattern that also appears as a suffix of the first *x* characters in the pattern. If there is no such prefix, then we know that we can skip over *x* characters. It is extremely important to note that the text itself is not considered if we have already matched *x* characters in the text, and then we know what that text is in terms of the pattern In the worst-case Knuth-Morris-Pratt algorithm we have to examine all the characters in the text and pattern at least once.

Boyer Moore algorithm scans the characters of the pattern from right to left beginning with the right most character. During the testing of a possible placement of pattern P against text T, a mismatch of text character T[i] = c with the corresponding pattern character P[j] is handled as follows:

- If C is not contained any were in P, then shift the pattern P completely past T[i]
- Otherwise, shift P until an occurrence of character C in P gets aligned with T[i].

The major drawback is the significant pre-processing time to set up the tables.

Some of the exact string matching algorithm used to solve the problem of searching and pattern in text are Boyer-Moore [5], Horspool[14], Zhu Takaoka, Naïve, Naïve FLC, and Naïve FML [4]. Although Horspool algorithm has better worst case running time than Boyer Moore. The latter is known to be extremely efficient in practice [2]. There have been many papers published that deal with exact pattern matching and introduce variants of Boyer-Moore algorithm

3.2 Common Issues of Existing Algorithms

Some of the common issues identified from the existing algorithms listed below

- The case sensitiveness is not considered
- Character wise searching is performed.
- Special characters and spaces are not eliminated.
- Pre-processed text is not formatted.

To overcome these issues and to speed up the searching a new algorithm is proposed. The spaces, special characters and stop words are not considered while searching. Pre-processing the text helps to minimize the size of search data. The reduced data is then stored in a data base where the searching actually done

4. PROPOSED WORK

Searching process of the text analyser tool is done in the proposed single pattern matching algorithm which is fast compared to existing single pattern matching algorithms. And the comparisons based on time complexity is given in this paper.

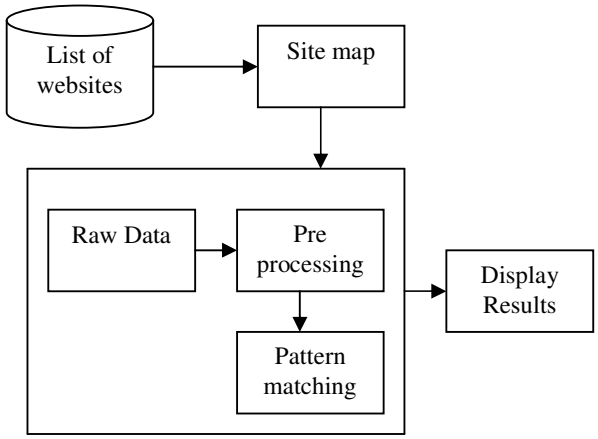

Figure 4.1: Text Analyser Flow Diagram

4.1 Site Map

This is the first phase in text analyser tool. Before searching process starts the Pre processing is done. Using web crawler a sitemap is developed. User needs to give a website and search query, and then the website is crawled to fetch all the hyperlinks available in it. Web crawlers are programs that give graph structure of the web to move page by page. Web crawlers are used to retrieve web pages and then to a local repository.

4.2 Pre processing

Pre-processing phase foresees the creation of a repository of the information from the web during the execution of the first phase. During this phase, the documents collected by the web crawlers will be pre-processed for converting the semi-structured data into a structured format. When parsing a webpage to extract content information it is necessary to remove commonly used words or stop words such as “it” ,”can”, “then” etc and this process is called stop listing. In addition to stop word elimination, stemming is done in order to normalize words to its root form. Eg: ”connected”, ”connecting”, ”connection” is all reduced to its root connect. This is done by porter stemming algorithm. Finally, to avoid type conflict while searching all upper case letters are converted into lower case.

4.3 Pattern Matching

Once the pre processing is done pattern matching is performed to find the word match. This is done by checking the length of the pattern and the text. If the length of the pattern and the text is matched further pattern matching is done with proposed pattern matching algorithm given below.

4.3.1 Proposed Pattern Matching Algorithm

- If the first character in the pattern is matched with first character in text, then mid characters are compared, if matches, then last character of pattern are compared with text.
- If all the three comparisons are succeeded then remaining characters in pattern are compared with text from left to right.
- If any mismatch occurs in any of the comparison then next occurrence of first character in pattern is searched in the text and search starts from that position.
- Other words are skipped without comparison.

4.3.2 Features of proposed algorithm

The following are the salient features of proposed algorithm

- Case insensitive
- Reduced time complexity
- Calculating the length of the words helps to skip unwanted words.
- Number of comparisons which affects the processing time, rapidly decreased after the FML match.
- By building the site map all the hyperlinks can be searched.

4.3.3 Algorithm for keyword searching

```
Algorithm PMA (String w ,String p)
    if (length (w) ==length (p))
    {
            if(w[0]=p[0]&&w[mid]=p[mid]&&
              w[last]=p[l ast])
    {
            Procedure1 (1,mid-1);
            Procedure1 (mid+1,last-1);

    }
     }
   Procedure1 (int a, int b)
    {
            For i=a to b do
                Repeat
                    If (w[i]!=p[i])
                    Return False
                Until True
    }
```

4.3.4 Example

Text: Due to this huge size of data, web search engines are becoming increasingly important as the primary means of locating relevant information. Such search engines rely on massive collections of web pages that are acquired with the help of web crawler, also known as robots or spiders. Crawler traverse the web by following hyperlinks and storing downloaded pages in a large database that is later indexed for efficient execution of user queries. They are the essential components of all search engines and becoming increasingly important in data mining and other indexing applications crowlar.

Pre-processed Text: huge size data web search engine important locate relevant inform. search engine massive collection web page acquire web crawler robots spiders Crawler traverse web follow hyperlink storing downloaded page database later index efficient execution user query essential components search engine become increase important data mine index applicant crowlar.

Pattern: **Crawler**

Step 1: length of the pattern = 7

Step 2: selects words from text of length 7

Text: Engines primary engines crawler crawler storing queries engines crowlar
Pattern: Crawler

Step 3: performs FML evaluation

Text: crawler crawler crowlar
Pattern: Crawler

Step 4: character matching is done

Text: Crawler Crawler
Pattern: Crawler

Step 5: Result: 2 words

4.3.5 Analysis

The length of the words equal to the length of the pattern is taken for searching. Remaining words are skipped. Considering there are **x** tokens of same length, FML (First Middle Last) characters search is performed, from which **n** tokens satisfies the FML condition. Then for each word the algorithm will not compare first, middle, last characters since they already matched at the previous step. So at each word algorithm compares **a-3** character until it reaches **n-1** tokens. So, this means the algorithms takes time complexity of **n (a-3**) for searching. Were **n** is number of tokens and **a** is length of token.

5. PERFORMANCE EVALUATION

5.1 Comparisons with Existing Algorithm

The algorithm was implemented using Java and JSP. It was tested using different file sizes. The performance of proposed algorithm is measured by comparing with existing algorithms such as Boyer Moore, Zhu Tahoka, Horspool, Naïve, NaïveFL, and Naïve FML. The comparison chart is shown below. The chart shows the reduced time complexity of the proposed algorithm

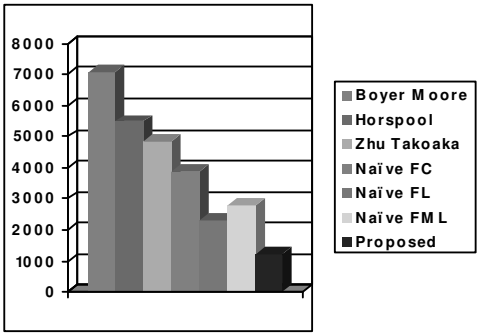

Fig 5.1 A Sample Execution Time

Fig-3 shows the experimental results of the tested algorithms. The average execution time of given keywords searched by various algorithms are plotted in above graph (Fig-3). The execution time in milliseconds is denoted in y-axis and algorithms taken for comparisons are denoted in x-axis. The values plotted in the above graph (Fig-3) are taken by finding average searching time of words in a file. The number of character comparisons can be used as a measuring factor, this factor affects time complexity. When number of character comparisons decreased complexity also decreased.

5.2 FML Condition Evaluation

After the preprocessing phase tokenized text file is generated and gives out the text file for pattern matching phase. The proposed pattern matching algorithm checks the length of the given search keyword and length of the word in the tokenized text file. If length doesn't match, next word in the tokenized text file is checked for same length. Once the length of keyword is matched with the word in the tokenized text file, First Middle Last characters are checked. This condition helps to skip large number of words in the tokenized text file without complete checking of all characters of a word. By checking only 3 characters of a word, it is found that if the words returns true for FML condition then the word to be searched are almost equal or same word. This is shown in below graphs Fig 5.2.1, Fig-5.2.2, Fig-5.2.3. The character comparison is reduced with the help of this condition, which is major measuring factor for time complexity. To show the efficiency of FML condition, a survey is done by collecting all words starting A to Z of length 10,9,8,7,6 and 5 is taken in text file and checked for average number of words that satisfy the FML condition. The graph below shows only words of length 10, 8 and 9.

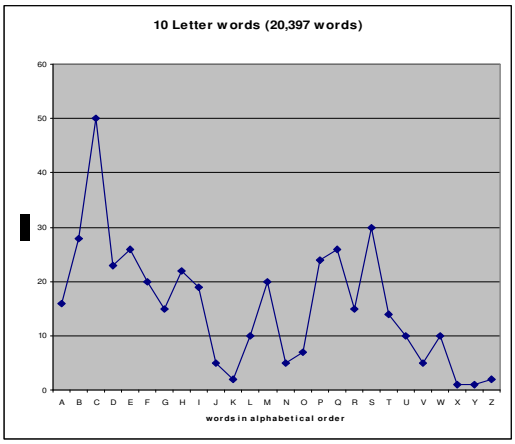

Fig-5.2.1: No. of words with length 10 satisfies FML condition

.

Fig-5.2.1 says that there is average of 52 words totally of length 10 with starting letters a-z satisfies FML condition among 20,397 words of length 10

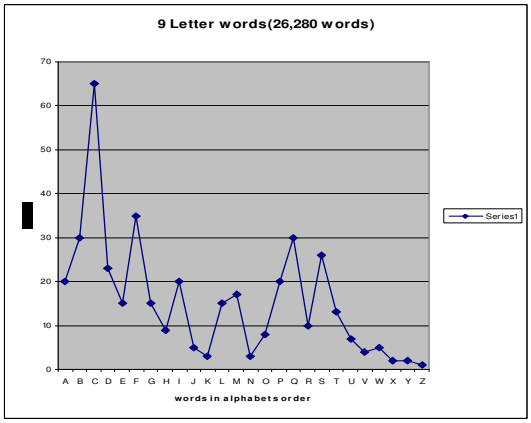

Fig-5.2.2: No. of words with length 9 satisfies FML condition

Fig-5.2.2 says that there is average of 65 words totally of length 9 with starting letters a-z satisfies FML condition among 26,280 words of length 9

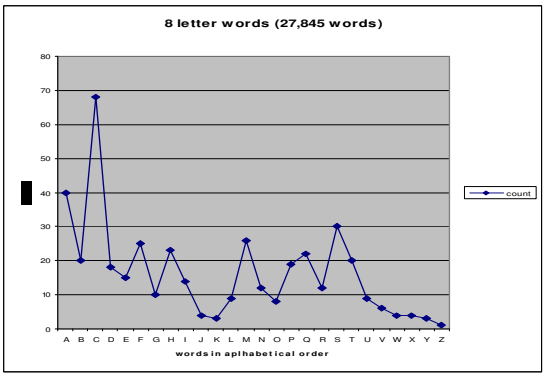

Fig-5.2.3: No. of words with length 8 satisfies FML condition

Fig-5.2.3 says that there is average of 68 words totally of length 8 with starting letters a-z satisfies FML condition among 27,845 words of length 8. This shows the efficiency of FML condition. Number of words returned true is very small compare to total number of words. Hence FML condition is checked before character by character matching.

6. ACKNOWLEDGEMENTS

This paper is a part of SSE project, Funded by National technical research organization, New Delhi.

7. CONCLUSIONS

To facilitate fast retrieval preprocessing is done from the collected information using certain technique called FML condition which is used to retrieve the needed words from the list of large set of words. A new exact pattern matching algorithm is proposed. All the algorithms including proposed algorithm, naïve, Boyer Moore, Horspool, and Zhu Tahoka have been implemented. Compared results and chart prove the performance enhancement and improved time complexity of the proposed algorithm.

8. REFERENCES

[1] Hai Dong, Farookh Khadeer Hussain, Elizabeth chang, (2008) State of art in metadata abstraction crawlers, IEEE computer society.
[2] Musbah Aqel & Ibrahiem.M, (2007)Multiple skip multiple pattern matching Algorithm, IAENG international journal.
[3] Qing Cheng Li, Shan Lin, Zhen hua dong, (2008) Research on web information mining by using Crawler techniques, IEEE computer society.
[4] Rami H.Mansi and jehad Q.odeh, (2009) Improving Naïve String matching algorithm, medwell journals.
[5] Robert S. Boyer & J Strother Moore,(1977) A fast String Searching algorithm, Communications of ACM
[6] Vishal Gupta, Lal Quan, (2008) A keywords searching for fast search engines, Ghaziabad IEEE computer society.
[7] Xiang peisu, tain ke, hunang Qinzhen, (2008)A framework of deep web crawler, IEEE computer society
[8] Xiaoming Song, Jianhua Feng Guoling Li,Qin Hong, (2008) LCA based keyword search for effective retrieval of "information unit" from web pages, IEEE computer society
[9] Yan Li, Xin Zhong Chen,Bing Ru Yang, (2002) Research on web mining based intelligent search engine, IEEE computer society
[10] Zu-Kuan Wel, Jiang-Ping Du, (2008)Research on several key issues about search engines, IEEE computer society
[11] http://wiki.answers.com/q/how_many_websites_are_there_in_the_www
[12] Zu-Hao Pan , Richard Chia-Tung Lee, (2008) An approximate String matching based on Exact String matching with constant pattern length, Department of computer science and Information Engineering, National Chi Nan university .
[13] Kunth, D.E, Morris J.H, Pratt V.R (1977), Fast pattern matching in strings, Journal on computing.
[14] Horspool R.N, (1980) Practical fast searching in strings, Software-practice & experience 10(6), pp 501-506
[15] K-W Liu, Richard Chia-Tung Lee, (2008) Some Results on Exact String matching algorithm, Department of computer science and Information Engineering, National Chi Nan university.