String Matching Algorithms For Reteriving Information From Desktop – Comparative Analysis

Dr. S.Vijayarani
Assistant Professor,
Department of Computer Science,
Bharathiar University,
Coimbatore.
vijimohan2000@yahoo.com,

Ms. R.Janani
PhD Research Scholar,
Department of Computer Science
Bharathiar University,
Coimbatore
janani.sengodi@gmail.com

Abstract- Information retrieval is the method of retrieving the knowledge relevant to an issue of curiosity. It locates the relevant documents, on the premise of user's question which includes keywords or example files. Probably the most acquainted application of information retrieval system is search engine corresponding to Web search, Desktop search, Federated search, Mobile search, Enterprise search and Social search. This research work mainly focused on the desktop search. Desktop search is the specified variant of enterprise search, where the information foundations are the files stored on a personal computer, together with email and websites established on content analysis. Content Analysis is a group of manual or computer based approaches for contextualized explanations of documents. To analyze the content the different text pattern matching algorithms are used and it is used to discover all the existences of a limited set of patterns inside an input document. Commonly these algorithms are used in several applications that include information security bio-informatics, plagiarism detection, text mining and document matching. String matching is essential for finding text patterns that are in online and offline. String matching algorithm is used to matches the pattern precisely or about in the input document. The main objective of this research work is to analyze the performance of existing string matching algorithms. For this comparison there are four algorithms are used namely, Two way algorithm, Colussi algorithm, Optimal mismatch algorithm and Maximal shift algorithm. From this analysis it is observed that the Colussi string matching algorithm gives the better result.

Keywords: String matching, Desktop Search, Two way algorithm, Colussi algorithm, Optimal mismatch algorithm, Maximal shift algorithm

I. INTRODUCTION

Information Retrieval system is used to identify the relevant information in a document database which matches a user's query. The text is considered to be composed of two important units in information retrieval system, specifically the document such as book, magazine paper, chapters, sections, paragraphs, web pages, computer source code, and so on, and the term corresponding to word, word-pair, and phrase within a document[1]. The main objective of this research work is to analyze the performance of existing string matching algorithm for desktop search. Desktop search is used to implements the searches over the content of the file or document. String matching algorithm is an essential idea for many problems and it is used in various applications which include text mining, data retrieval, DNA pattern matching and finding certain vital keywords in security applications. Strings matching algorithms has two strategies such as, exact matching and approximate matching [2]. In exact string matching, the pattern is completely matched with the specific text window of input text and it displays the initial index position. In approximate string matching, if precise portion of the pattern matched with the selected text window straight away it displays the output. This paper organized as follows section II describes the literature survey; section III illustrates the methodology of this research work. Result and discussion given in Section IV and section V describes the conclusion of this research work.

II. RELATED WORK

Chinta Someswara Rao et al [12] implemented parallel string matching with JAVA Multithreading with multi core processing, and performed a comparative study on Knuth Morris Pratt, Boyer Moore and Brute force string matching algorithms. For testing, gene sequence database are used which consists of lacks of records. From the test results it is shown that the multi core processing is better compared to lower versions. Finally this proposed parallel string matching with multi core processing is better compared to other sequential approaches.

Abdulwahab Ali Al-mazroi and Nur'Aini Abdul Rashid [4] proposed a new hybrid algorithm called BRSS by combining two algorithms, Berry-Ravindran and Skip Search. The hybrid algorithm demonstrates enhanced character comparisons, number of attempts and searching time performances in all the different data size and pattern lengths, therefore the proposed algorithm is useful for searching DNA, Protein and English text. This also proved that the application of the hybrid algorithm will lead to better searching and matching of the patterns than the use of one algorithm as data is becoming more complex presently.

Saima Hasib, Mahak Motwani and Amit Saxena [5] discussed the Aho-Corasick algorithm is best suited for multiple pattern matching and it can be used in many application areas. The complexity of the algorithm is linear in the length of the patterns plus the time taken of the searched text plus the amount of output matches. It is found to be attractive in large numbers of keywords, since all keywords can be simultaneously matched in one pass. Aho-Corasick provides solution to many real world problems like Intrusion detection, Plagiarism detection, bioinformatics, digital forensic, text mining and many more. Aho-Corasick is one of the most productive algorithms in text mining.

Jorma Tarhio and Esko Ukkonen [7] proposed an efficient string matching algorithm (named ACM) with compact memory as well as high worst-case performance. Using a magic number heuristic based on the Chinese Remainder Theorem, the proposed ACM significantly reduces the memory requirement without bringing complex processes. Furthermore, the latency of off-chip memory references is drastically reduced. The proposed ACM can be easily implemented in hardware and software. As a result, ACM enables cost-effective and efficient IDSs.

III. Methodology

The main objective of this research work is to analyze the performance of existing string matching algorithms. In order to perform this task, this research work uses four existing string matching algorithms; Two way algorithm, Colussi algorithm, Optimal mismatch algorithm and Maximal shift algorithm. The performance factors are used time taken for searching the pattern, number of iterations required and its accuracy for single word search, multiple words search and a file search. Figure 1 represents the architecture of this research work.

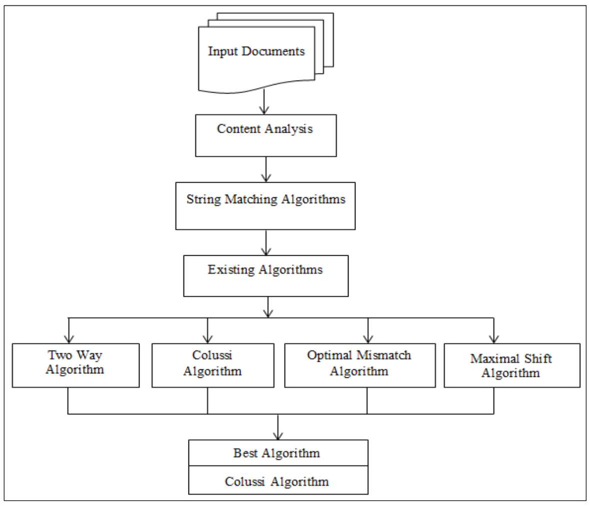


Fig. 1. System Architecture

A. Two Way Algorithm

The Two Way algorithm is the efficient string matching algorithm. The pattern P is factorized into two parts P_i and P_r such that the pattern $P=P_iP_r$. At that time the searching phase of this algorithm involves in matching the characters of P_r from left to right. If no mismatch occurs during the first stage, in comparing the characters of P_i from right to left in a second stage. In this case when a mismatch occurs while comparing the k-th character of P_r, then a shift of length k can be performed [3]. When a mismatch occurs while comparing P_r, or when an occurrence of the pattern is found, then shift of length per (P) (critical factorization) can be performed.

The searching phase of this algorithm can be done in O (n) time complexity. The main features of this algorithm are it requires a proper alphabetical order and it performs 2n-m text character comparisons in the worst case complexity.

Algorithm 1. *Two Way Algorithm*

```
Declare the variables i, j, ell=i, memory, p, per, q;
if (memory(x, x + per, ell + 1) == 0) then
    Initialize j = 0; memory = -1;
While (j <= n - m) then
        Initialize i = MAX (ell, memory) + 1;
While (i < m && x[i] == y[i + j])
            Increment the i value
If (i >= m) then
            Assign i = ell;
While (i > memory && x[i] == y[i + j])
                Decrement the i value
If (i <= memory)
Print the result
            j += per;
Memory = m - per - 1;
Else j += (i - ell);
Memory = -1;
Else calculate per = MAX (ell + 1, m - ell - 1) + 1;
    j = 0;
While (j <= n - m) {
        i = ell + 1;
While (i < m && x[i] == y[i + j])
            ++i;
Print "Pattern Found"
```

B. Colussi Algorithm

The major design of Colussi algorithm follows a constricted analysis of Knuth-Morris-Pratt (KMP) algorithm. In this algorithm, the set of pattern positions can be divided into two disjoint subsets [5]. Then each and every attempt involves in the two phases:

Phase 1: In this phase the comparisons can be performed from left to right with respect to the text characters aligned with the pattern position for which value of the function is exactingly greater than -1. This position is called noholes position.

Phase 2: This phase consists of comparing the remaining positions, which is called holes position from right to left order.

The main advantage of this algorithm, when a mismatch occurs during the phase 1, after the suitable shift it is not necessary to compare the text characters aligned with the noholes position. When a mismatch occurs during the phase 2, it means that a suffix of the pattern matches a factor of text after the consistent shift of prefix of the particular pattern [6]. The searching phase of this algorithm can be done in O (n) time complexity and it performs the 3/2 n comparison during the search phase.

Algorithm 2. *Colussi Algorithm*

```
Pattern- pat; Character- c, Location –loc;
Length-m, Number of characters-n;
Initialize pat=0, i=0,j=0;
Colussi(x, m, h, next, shift)
   Last = -1;
   While (j <= n - m) then
   While (i < m && last < j + h[i] &&
   x [h[i]] == y[j + h[i]])
   i++;
   If (i >= m || last >= j + h[i]) then
   Print the output
         i = m;
   If (i >nd)
   Last = j + m - 1;
       j += shift[i];
       i = next[i];
```

```
Shift Calculation
    For (i = 0; i <= nd; ++i)
    Shift[i] = kmin [h[i]];
    For (i = nd + 1; i < m; ++i)
    Shift[i] = rmin [h[i]];
    Shift[m] = rmin [0];
Next Calculation
    For (i = 0; i <= nd; ++i)
    Next[i] = nhd0 [h[i] - kmin [h[i]]];
    For (i = nd + 1; i < m; ++i)
    Next[i] = nhd0 [m - rmin [h[i]]];
    Next[m] = nhd0 [m - rmin [h[m - 1]]];
    Return (nd);
h Value Calculation
    s = -1;
    r = m;
    For (i = 0; i < m; ++i)
    If (kmin[i] == 0)
    h [--r] = i;
    Else
    h [++s] = i;
    Nd = s;
```

C. Optimal Mismatch Algorithm

The optimal mismatch algorithm is the variant of the quick search algorithm, where the pattern characters are compared from the least frequent character one to most frequent character [13]. If the mismatch occurs most of the times but this algorithm checks the whole input text very quickly. Here one character needs to know the frequencies of each character of the alphabets. In this algorithm the preprocessing phase consists sorting the pattern character in decreasing order of their frequencies. And then build the quick search bad character shift and good suffix shift function modified to the order of pattern characters [9]. The preprocessing phase can be done in O (m^2+sigma) time complexity and O(m+sigma) space complexity. The searching phase of this algorithm can be done in O (mn) time complexity.

Algorithm 3. *Optimal Mismatch Algorithm*

```
Int i, j, Gs- good suffix, qsBc-quick search bad character;
Pattern pat, m- length of pattern, int i;
For (i = 0; i <= m; ++i) then
Pat[i].loc = i;
Pat[i].c = x[i];
  j = 0;
While (j <= n - m) then
     Assign i = 0;
While (i < m && pat[i].c == y[j + pat[i].loc]) then
       Increment i value
If (i >= m)
Output
     j += MAX (adaptedGs[i], qsBc[y [j + m]]);
```

D. Maximal Shift Algorithm

The maximal shift algorithm is the modification of quick search algorithm, where the pattern characters are compared from the one which will lead to a larger shift to the one which will lead to a shorter shift. In this algorithm the preprocessing phase consists sorting the pattern character in decreasing order of their shift. And then build the quick search bad character shift function and good suffix shift function modified to the order of pattern characters [9] [10]. The preprocessing phase can be done in O ($m^2+\pi$) time complexity and O ($m+\pi$) space complexity. The searching phase of this algorithm has a quadratic worst case time complexity.

Algorithm 4. *Maximal Shift Algorithm*

```
Pattern- pat; Minimum Shift- minsft; Maximum Shift- maxsft
Character- c, Location –loc, m-length;
 Number of characters-n;
Initialize pat=0, i=0,j=0;
Compute the minsft and maxsft values
While (j<=n-m) then
        Initialize i=0;
While (i < m && pat[i].c == y[j + pat[i].loc]) then
        Increment the i value
if (i >= m) then
       Print "Pattern Not Found"
       j += MAX (adaptedGs[i], qsBc[y[j + m]]);
Else
       Print "Pattern Found"
```

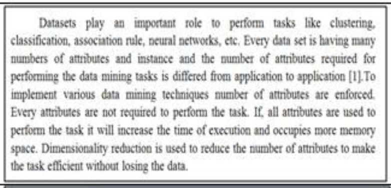
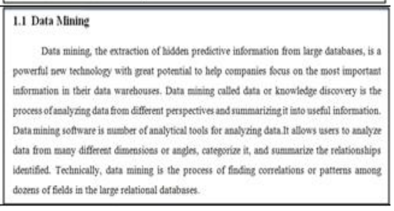
IV. RESULT AND DISCUSSION

In order to perform this analysis, the performance factors are search time, number of iterations and relevancy for various types of inputs. The inputs are single word, multiple words and a file. For this analysis, the existing string matching algorithms were implemented by using Java.

- Search Time: It refers the time taken for searching the pattern within the input text. It can be estimated by comparison of each character in pattern with the input text.
- Iterations: It refers the total number of iterations for matching the pattern with the input text. It is based on the given input document and various algorithms.
- Relevancy: It refers the accuracy of the algorithm; the accuracy is calculated by using the formula as follows,

$$\boxed{\text{Accuracy}=\frac{\textit{Total number of patterns reterived}}{\textit{Total number of patterns in text}}\text{ X }100}$$

TABLE 1: Sample Input

File Name	Number of Words	Size (KB)	Sample
mine.txt	1567	10.41	 Datasets play an important role to perform tasks like clustering, classification, association rule, neural networks, etc. Every data set is having many numbers of attributes and instance and the number of attributes required for performing the data mining tasks is differed from application to application [1].To implement various data mining techniques number of attributes are enforced. Every attributes are not required to perform the task. If, all attributes are used to perform the task it will increase the time of execution and occupies more memory space. Dimensionality reduction is used to reduce the number of attributes to make the task efficient without losing the data.
pro.docx	2123	22.6	 **1.1 Data Mining** Data mining, the extraction of hidden predictive information from large databases, is a powerful new technology with great potential to help companies focus on the most important information in their data warehouses. Data mining called data or knowledge discovery is the process of analyzing data from different perspectives and summarizing it into useful information. Data mining software is number of analytical tools for analyzing data.It allows users to analyze data from many different dimensions or angles, categorize it, and summarize the relationships identified. Technically, data mining is the process of finding correlations or patterns among dozens of fields in the large relational databases.

The table 2 illustrates the performance metrics like time, number of iterations and relevancy of Two Way Algorithm for text file (mine.txt).

TABLE 2: Performance analysis of Two Way Algorithm for text files (mine.txt)

Input	**Performance Measure**		
	Time (ms)	***Number of Iterations***	***Relevancy (%)***
Single Word	2	5	100
Multiple Words	26	29	98
File	64	58	89

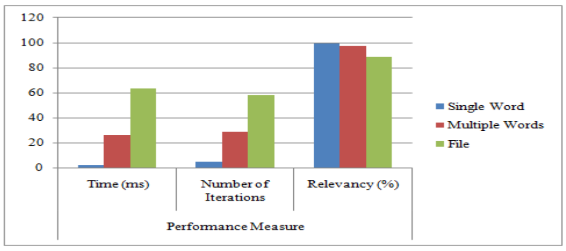


Fig 2. Performance analysis of Two Way Algorithm

The table 3 describes the performance measures like time, number of iterations and relevancy of Two Way Algorithm for docx file (pro.docx).

TABLE 3: Performance analysis of Two Way Algorithm for docx files (pro.docx)

Input	Performance Measure		
	Time (ms)	Number of Iterations	Relevancy (%)
Single Word	5	9	100
Multiple Words	31	35	97
File	75	64	89

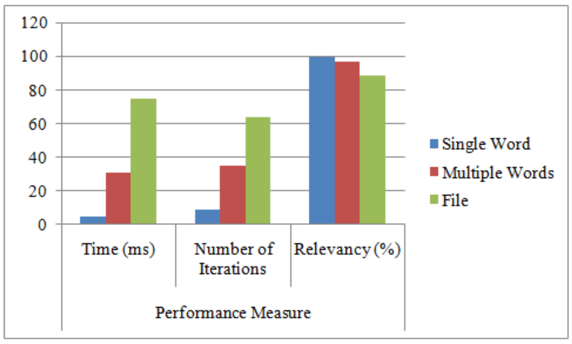


Fig 3. Performance analysis of Two Way Algorithm

Table 4: Performance analysis of Colussi Algorithm for text files (mine.txt)

Input	Performance Measure		
	Time (ms)	Number of Iterations	Relevancy (%)
Single Word	2	3	100
Multiple Words	19	20	98
File	53	51	91

The table 4 describes the performance measures like time, number of iterations and relevancy of Colussi algorithm for text file (mine.txt).

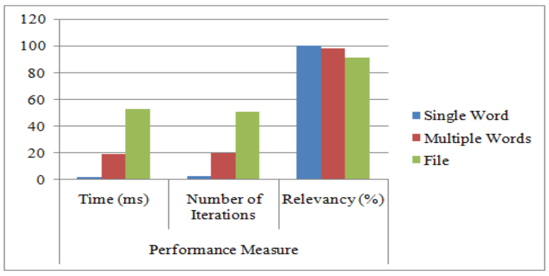


Fig 4. Performance analysis of Colussi Algorithm

The table 5 describes the performance measures like time, number of iterations and relevancy of Colussi algorithm for docx file (pro.docx).

TABLE 5: Performance analysis of Colussi Algorithm for docx files (pro.docx)

Input	Performance Measure		
	Time (ms)	Number of Iterations	Relevancy (%)
Single Word	4	6	100
Multiple Words	26	30	99
File	60	51	92

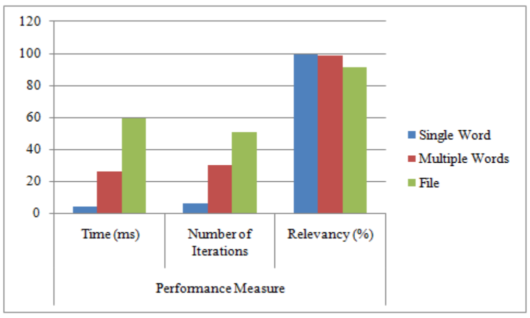


Fig 5. Performance analysis of Colussi Algorithm

TABLE 6: Performance analysis of Optimal mismatch Algorithm for text files (mine.txt)

Input	Performance Measure		
	Time (ms)	Number of Iterations	Relevancy (%)
Single Word	6	5	100
Multiple Words	29	27	98
File	96	59	84

The table 6 describes the performance measures like time, number of iterations and relevancy of Optimal Mismatch algorithm for text file (mine.txt).

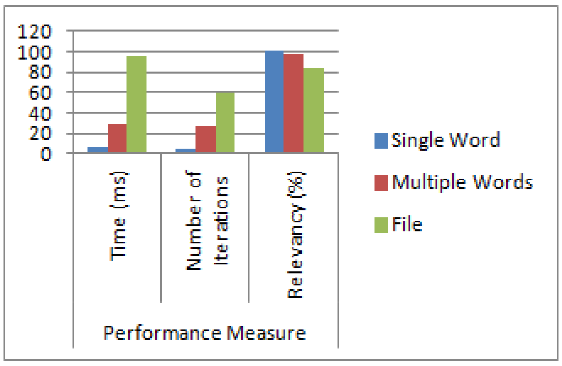


Fig 6. Performance analysis of Optimal Mismatch Algorithm

The table 7 describes the performance measures like time, number of iterations and relevancy of Optimal Mismatch algorithm for docx file (pro.docx).

TABLE 7: Performance analysis of Optimal Mismatch Algorithm for docx files (pro.docx)

Input	Performance Measure		
	Time (ms)	Number of Iterations	Relevancy (%)
Single Word	9	14	99
Multiple Words	36	30	99
File	82	60	81

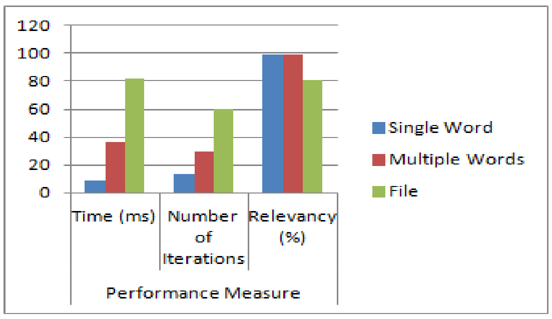


Fig 7. Performance analysis of Optimal Mismatch Algorithm

TABLE 8: Performance analysis of Maximal Shift Algorithm for text files (mine.txt)

Input	Performance Measure		
	Time (ms)	Number of Iterations	Relevancy (%)
Single Word	5	9	100
Multiple Words	25	30	94
File	98	58	91

The table 8 describes the performance measures like time, number of iterations and relevancy of Maximal Shift algorithm for text file (mine.txt).

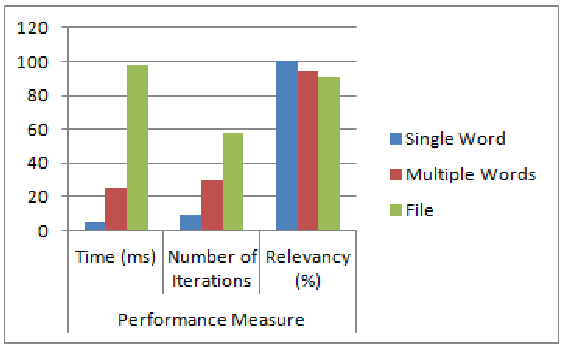


Fig 8. Performance analysis of Maximal Shift Algorithm

The table 9 describes the performance measures like time, number of iterations and relevancy of Maximal Shift algorithm for docx file (pro.docx).

TABLE 9: Performance analysis of Maximal Shift Algorithm for docx files (pro.docx)

Input	Performance Measure		
	Time (ms)	Number of Iterations	Relevancy (%)
Single Word	6	7	99
Multiple Words	19	29	97
File	51	67	86

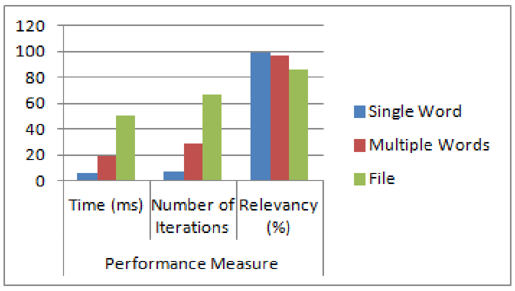


Fig 9. Performance analysis of Maximal Shift Algorithm

V. Conclusion

Information retrieval (IR) is used to discover the significant documents in a large collection of documents which is matching a user's query. The main goal of information retrieval System is to identify the significant information that satisfies the user information needs. To analyze the content the various string matching algorithms are used and it is used to discover all the existence of a limited set of patterns inside an input document. String matching algorithm is used to matches the pattern exactly or approximately within the input document. This research work analyzes the performance of existing string matching algorithms namely, Two way algorithm, Colussi algorithm, Optimal mismatch algorithm, and Maximal shift algorithm. From this analysis it is observed that the Colussi algorithm gives better accuracy for the various inputs.

REFERENCES

[1]. Mahmoud Moh'dMhashi , Mohammed Alwakeel, New Enhanced Exact String, Searching Algorithm, IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.4, April 2010

[2]. Christian Charras, Thierry Lecroq and Joseph Daniel, A Very fast string searching algorithm for small alphabets and long patterns, Combinational Pattern Matching, 9th Annual Symposium, CPM 98 Piscataway, New Jersey, USA, 2005

[3]. R.S. Boyer, J.S. Moore, "A fast string searching algorithm," Communication of the ACM, Vol. 20, No. 10, 1977, pp.762– 772.

[4]. Abdulwahab Ali Al-Mazroi and Nur'aini Abdul Rashid, A Fast Hybrid Algorithm for the Exact String Matching Problem, American Journal of Engineering and Applied Sciences 4 (1): 102-107, 2011.

[5]. Ababneh Mohammad, OqeiliSaleh and Rawan A Abdeen, Occurrences Algorithm for String Searching Based on Brute-Force Algorithm, Journal of Computer Science, 2(1): 82-85, 2006.

[6]. Bin Wang, Zhiwei Li, Mingjing Li and Wei-Ying Ma, Large-Scale Duplicate Detection for Web Image Search, Multimedia and Expo, IEEE International Conference, 353-356, 2006

[7]. Pandiselvam.P, Marimuthu.T ,Lawrance. R, A Comparative Study On String Matching Algorithms Of Biological Sequences.

[8]. JormaTarhio and EskoUkkonen, Approximate Boyer-Moore String Matching, SIAM Journal on Computing, Volume 22 Issue 2, 243 – 260, 1993.

[9]. Olivier Danvy, Henning Korsholm Rohde, On Obtaining the Boyer-Moore String-Matching Algorithm by Partial Evaluation, Journal of Information Processing Letters, Volume 99 Issue 4, 158-162, 2005.

[10]. Robert S. Boyer and J. Strother Moore, A fast string Searching Algorithm, Communication of the ACM, Volume 20 Issue 10, 762-772, 1977.

[11]. Akinul Islam Jony, Analysis of Multiple String Pattern Matching Algorithms, International Journal of Advanced Computer Science and Information Technology (IJACSIT) Vol. 3, No. 4, 2014, Page: 344-353, ISSN: 2296-1739

[12]. Ashish Prosad Gope , Rabi Narayan Behera, A Novel Pattern Matching Algorithm in Genome Sequence Analysis, (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (4) , 2014, 5450-5457

[13]. SonawaneKiranShivaji, Prabhudeva S, Plagiarism Detection by using Karp-Rabin and StringMatching Algorithm Together, International Journal of Computer Applications (0975–8887)Volume 116 – No. 23, April 2015

[14]. Mustafa Abdul Sahib Naser, Nur'Aini Abdul Rashid, and Mohammed FaizAboalmaaly, Quick-SkipSearch Hybrid Algorithm for the Exact String Matching Problem, International Journal of Computer Theory and Engineering Vol. 4, No. 2, April 2012

[15]. JamunaBhandari, Anil Kumar, String Matching Rules Used By Variants Of Boyer-MooreAlgorithm ,Journal of Global Research in Computer Science, Volume 5, No. 1, January 2014

[16]. Mr. Rahul, B. Diwate, Prof. Satish J. Alaspurkar, Study of Different Algorithms for Pattern Matching, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 3, March 2013

[17]. Akinul Islam Jony, Analysis of Multiple String Pattern Matching
Algorithms, International Journal of Advanced Computer Science and
Information Technology (IJACSIT) , Vol. 3, No. 4, 2014, Page: 344-353, ISSN: 2296-1739

[18]. Simon Wahlström, Evaluation of String Searching Algorithms,2004.