JURNAL INFORMATIKA
Vol. 14, No 1, January 2020, pp. *7-14*
ISSN: 1978-0524

Comparison of Knuth Morris Pratt and Boyer Moore algorithms for a web-based dictionary of computer terms

Ali Khumaidi[a,1,*], Yusuf Aras Ronisah[b,2], Harjono Padmono Putro[c,3]

[ab] Teknik Informatika, Universitas Krisnadwipayana, Jl. Jatiwaringin, Pondok Gede, Jakarta, Indonesia
[1] *alikhumaidi@unkris.ac.id*;* [2] *ayasyusuf42@gmail.com;* [3] *harjonopputro@unkris.ac.id*
** corresponding author*

ABSTRACT

Computer students need a dictionary of computer terms to deepen lectures. In developing dictionary applications, the term computer will choose the fastest and most efficient memory algorithm. The comparison algorithm is Knuth Morris Pratt (KMP) and Boyer Moore (BM) algorithm. Based on previous research, the KMP algorithm has a better performance compared to other string matching algorithms. However, other studies have concluded that the BM algorithm has better performance. Besides, the Zhu-Takaoka algorithm is more efficient than the KMP algorithm in dictionary development. The BM algorithm has the same search concept as the Zhu-Takaoka algorithm. The determination of the fastest and most efficient algorithm in this study uses the Exponential Comparison Method (ECM). ECM sets criteria for when searching and using the memory in the search process. The results of the comparison of the KMP and BM algorithm are the search time for the BM algorithm is 37.9%, and the KMP algorithm is 62.1%. The results of the use of search memory for the KMP algorithm are 50.6%, and the BM algorithm is 49.4%. The total ECM score shows that the BM algorithm is 0.55% better than the KMP algorithm.

Keywords:
String Matching
Knuth-Morris-Pratt
Boyer Moore
Exponential Comparison Method
Dictionary of Computer Terms

I. Introduction

The term computer dictionary is a collection of vocabulary words that have meanings and functions to guide human interaction with computers. The Dictionary of computer terms is currently widely distributed in printed books but is not efficient in its use. Users have to search for terms manually page by page to get the meaning of the words searched. Besides, printed books are also inflexible in their use and are not updated with the latest provisions.

Users can use online applications to search for computer terms by utilizing web applications such as https://kbbi.kata.web.id and http://intisari.grid.id as well as search engines. However, the time is sometimes not found or obtained a different meaning from the term computer. Even the current activities in searching vocabulary and dictionary meanings of computer terms are less active. The search method is still conventional, with applications that have many categories of words that are not computer-specific. Thus, the search process is too long.

Research related to string matching algorithm in documenting content comparing Brute Force algorithm, Knuth-Morris-Pratt algorithm (KMP), Boyer Moore algorithm (BM), and Rabin Karp algorithm. Using the KMP algorithm has a better speed than the others [1]. Besides, the application of the .txt and .docx document search algorithm shows that the KMP algorithm has the best performance [2]. Full permutation pattern-matching research with BM, Harspool, Aho-Corasick, and KMP algorithms concludes that the KMP algorithm has the fastest permutation pattern-matching performance [3]. However, a comparison of the performance of the KMP, Naïve, and BM algorithms in processing various text sizes concluded that the BM algorithm has the best and most efficient performance for large text sizes [4]. A word search in the dictionary with the Zhu-Takaoka and KMP algorithm is found that the Zhu-Takaoka algorithm is faster than the KMP algorithm [5]. The Zhu-Takaoka algorithm starts searching from the end of the pattern adapted to the text, but in the KMP algorithm, it starts from the beginning of the model. The BM algorithm has the same search concept as the Zhu-Takaoka algorithm doing a search starting from the end of the pattern. The application of the BM algorithm in the electronic term dictionary has a pretty good speed [6].

DOI: 10.26555/jifo.v14i1.a17038
W : http://journal.uad.ac.id/index.php/JIFO
E : jifo@uad.ac.id

Based on previous research, each string matching algorithm has the best conditions for different dictionary applications. This research will build a web application to compare the KMP and BM algorithms for the dictionary of computer terms. The initial hypothesis shows that the KMP algorithm is faster than the BM algorithm. The comparison process uses the Exponential Comparison Method (ECM). ECM is a decision support system method used to determine the priority order of decisions based on alternative criteria. ECM is quite useful in assessing performance [7, 8]. The determination of the algorithm in making a dictionary of computer terms is to choose the smallest total score. The attributes are word search time and memory usage in the search process.

II. Method

The stages of research comparing the KMP and Boyer Moore algorithms were carried out with the initial stages of observation and interviews. This stage is done by asking the method in finding a dictionary of computer terms and obstacles encountered. The author looks for facts directly by trying the processes carried out by users in searching for computer terms.

The next stage is the study of literature. This stage aims to find some documentation related to dictionary applications and string matching algorithms. Based on the results of previous research studies related to the comparison of the KMP and BM algorithms. The following will explain the concept of string matching and two algorithms.

String matching is a technique for finding patterns of character or strings [9]. String Matching is an algorithm that can search all short strings that appear with a pattern [0 .... n-1] and a longer one with a pattern [0 ... m-1] called text [10]. String searching can be formulated using two-terms, namely, the pattern (a string of length n), and text (a string of length m characters).

Character values (n<m) can be found in the text. In string matching algorithms, the text is defined to be in memory so that it is in the search string in the archive. The contents of the file are read first and stored in the memory [11]. If the pattern appears more than once in the text, then the search will bring up output in the form of the location of the model found first. There are many string matching algorithms, but in this study, only comparing the KMP algorithm and the Boyer Moore algorithm.

The KMP algorithm has an unnecessary iteration concept. The KMP algorithm process will not produce a match between the pattern or words sought with the primary model. For example, the process of finding some m in the sentence K[] containing the word k[]. The way it works is by looking for a match of characters at successive values starting from the m-index at the location of the string to be searched, that is, K [m]. If the index m is at the end of the string and no matching characters are found, the search will fail. For location m, the algorithm checks to find a match starting with the first character for the word to be searched, which is K[m] = k[0]? When a match is found, the algorithm will test other characters in the word search by checking the values in sequence starting from the location of the word index, i. The algorithm will take the character k [i] in the word search and check it according to the expression K[m+i] = k[i] ?. When there is a match between all sequential characters that correspond to locations k and m, the search string matches. The KMP algorithm application is implemented in desktop-based search engine applications [12].

The stages of the KMP algorithm in string matching are as follows:

1 The KMP algorithm starts to match the pattern at the beginning of the text.
2 The algorithm matches data from left to right for each character with a pattern in the corresponding text, until the following conditions (as appropriate):
   a. There is no match between character patterns and text (mismatch).
   b. There is a character pattern match. Then the algorithm will inform its position.
   c. The algorithm shifts the pattern according to the table—repeat step-b to the last text.

Pseudocode KMP algorithm in the pre-search and search phase as follows:

KMP algorithm in pre-search phase

```
Procedure preKMP(
  input P: array[0..n-1] of char,
  input n: integer,
  output kmpNext: array[0..n] of integer)

Declaration:
  i,j: integer

Algorithm
  i:= 0;
  j:= kmpNext[0]:= -1;
 while (i < n) {
    while (j > -1 and not(P[i] = P[j]))
      j:= kmpNext[j];
      i:= i+1;
      j:= j+1;
      if (P[i] = P[j])
        kmpNext[i]:= kmpNext[j];
      else
        kmpNext[i]:= j;
      endif
  endwhile
```

KMP algorithm in search phase

```
procedure KMPSearch(
  input m, n: integer,
  input P: array[0..n-1] of char,
  input : array[0..m-1] of char,
  output ketemu: array[0..m-1] of boolean)

Declaration:
  i, j,next: integer
  kmpNext: array[0..n] of interger

Algorithm:
  preKMP(n, P, kmpNext)
  i:=0
  while (i<= m-n) do
     j:=0
    while (j < n and T[i+j] = P[j]) do
       j:=j+1
     endwhile
     if(j >= n) then
       ketemu[i]:=true;
     endif
     next:= j - kmpNext[j]
       i:= i+next
  endwhile
```

Boyer Moore (BM) algorithm uses the concept of string matching from right to left. The process is done by scanning character patterns from right to left [13]. Boyer Moore's algorithm uses two-shift functions, namely good-suffix shift, and bad-character shift. This function is used to take the next step if there is a mismatch between text characters and character patterns [14].

The working principle of Boyer Moore's algorithm is as follows:

1. The preBmBc and preBmGs procedures will be performed to obtain initialization.
2. The results of the preBmBc and preBmGs procedures in the form of BmBc and BmGs tables are used for the string search process.

Pseudocode BM algorithm in the pre-search and search phase as follows:

BM algorithm in Pre-search

```
procedure preBmGs(
  input P : array[0..n-1] of char,
  input n : integer,
  input/output bmBc : array[0..n-1] of integer)
Declaration:
  i, j: integer
  suff: array [0..Alpabet] of integer
  preSuffixes(x, n, suff);
  for (i := 0 to m-1)
    bmGs[i] := n
  endfor
  j := 0
  for (i := n - 1 downto 0)
    if (suff[i] = i + 1)
      for (j:=j to n - 2 - i)
        if (bmGs[j] = n)
          bmGs[j] := n - 1 - i
        endif
      endfor
    endif
  endfor
  for (i = 0 to n - 2)
    bmGs[n - 1 - suff[i]] := n - 1 - i;
  endfor
```

BM algorithm in Search

```
procedure BMSearch(
  input m, n : integer,
  input P : array[0..n-1] of char,
  input T : array[0..m-1] of char,
  output ketemu : array[0..m-1] of boolean)
Declaration:
  i, j, shift, bmBcShift, bmGsShift : integer
  BmBc : array[0..255] of interger
  BmGs : array[0..n-1] of interger
Algorithm:
  preBmBc(n, P, BmBc)
  preBmGs(n, P, BmGs)
  i:=0
  while (i<= m-n)
    do j:=n-1
  while (j >=0 n and T[i+j] = P[j])
    do j:=j-1
  endwhile
  if (j < 0) then
    ketemu[i]:=true;
  endif
  bmBcShift:= BmBc[chartoint(T[i+j])]-n+j+1
  bmGsShift:= BmGs[j]
  shift:= max(bmBcShift, bmGsShift)
  i:= i+shift
```

Design and implementation of applications, build web-based applications using php, html, css, js and mysql databases. To illustrate the system, class diagrams and sequence diagrams are made.

Testing, comparing the score results from the KMP algorithm and Boyer Moore algorithm by measuring speed and memory usage based on ECM. To test the best algorithm, MPE is used to determine the priority order of alternative decisions with multiple criteria. In calculating and comparing the search process of the KMP and BM algorithm are as follows:

1. Determine alternatives

   To analyze the speed comparison between the KMP and BM algorithm in searching it is necessary to determine which algorithm will be used as a string search algorithm in a computer language dictionary application.

2. Determine Criteria

   To be able to compare the two alternatives, the next step is to determine criteria in analyzing the process and how it works. For the criteria, see table 1.

Table 1. Determination of criteria

Criteria	Description
Large memory used when searching	The calculation of memory usage occurs when the algorithm matches the string
The amount of time spent searching	The time calculation is obtained when the algorithm matches the string from the beginning of the match to the end

After determining the next criteria is the weighting of criteria as in table 2.

Table 2. Determination of criteria

Criteria	Influence of Criteria	Weight Range (0-1)	Description
Memory usage	50%	0,5	The level of memory usage has an effect on the algorithm's speed
Time Needed	50%	0,5	The time needed in the search process

III. Result and Discussion

A. The Need for a Dictionary of Computer Terms

The author conducted interviews with a sample of 50 students of Informatics Engineering study programs to determine the need for a computer term dictionary. The discussion is divided into three main points, namely the importance of the glossary of computer terms, how to access computer terms, and how difficult it is to obtain. Fig. 1 shows that all respondents stated that they needed access to a dictionary of computer terms to support the learning process. Respondents gave responses with the percentage of strongly agreeing at 86% and agreeing at 14%. Based on access, the dictionary of computer terms is divided into three, namely printed books, search engines, and KBBI or other web applications. Fig. 1 explained that of the three ways of access, respondents stated 88% experienced difficulties and were quite severe, while 12% said they were comfortable.

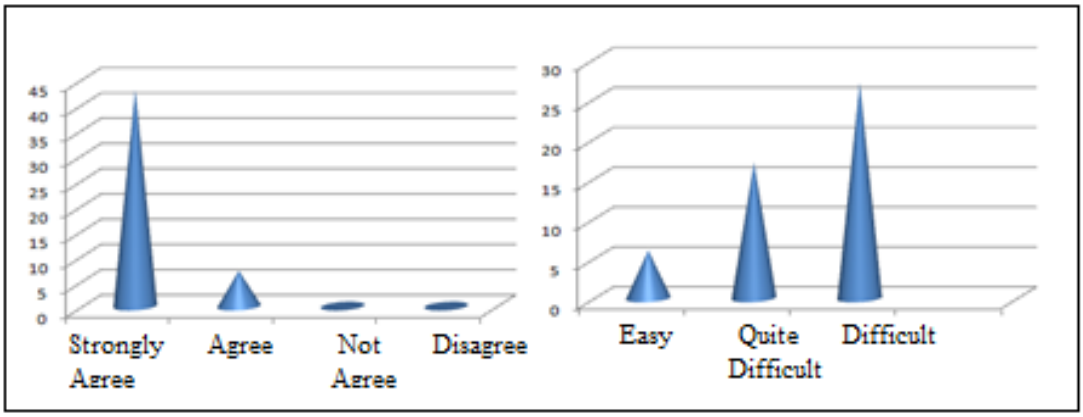

Fig. 1. Graph of needs access and difficulty to the dictionary of computer terms

After the interview, researchers observed the difficulty level of access to the dictionary of computer terms. Accessing a glossary of computer terms using books has many limitations, namely media and speed in searching. The use of online tools has difficulty in search results and seek time because not all application providers have keyword choices. Therefore, a computer term dictionary application with string matching is indispensable for an easy and fast term search.

B. Application design of algorithm comparison

The application is based on a website, which users can access the application using a browser. The app is designed with PHP, HTML, CSS, js, and MySQL database. This application compares the KMP and BM algorithm by searching for computer terms, and the search results will be obtained in the form of memory usage information and access time. Next, Fig. 2 shows a sequence diagram of the application being built.

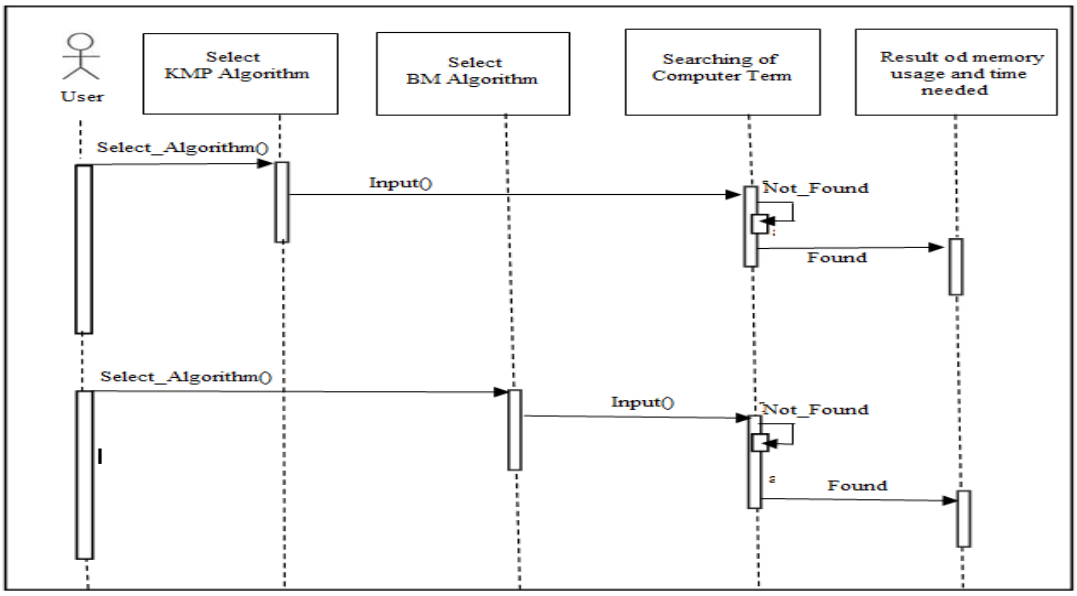

Fig. 2. Sequence diagram of application comparison of algorithms

The class diagram illustrates the interaction between classes in the application. There are three classes of this design, namely users, algorithms, and computer search terms. Based on Fig. 3, the design of this application to compare the KMP and BM algorithm.

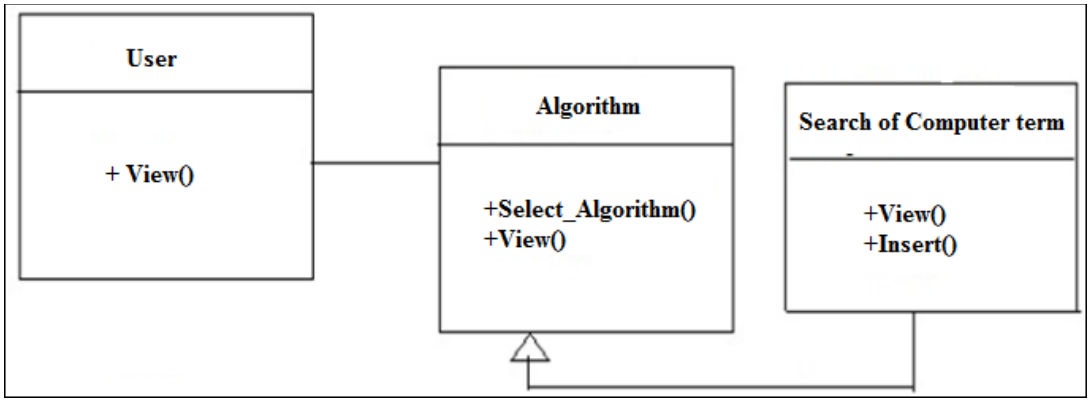

Fig. 3. Class diagram of application comparison of algorithms

Fig. 4 shows the main display and the results of a computer term search that can be accessed through a browser.

Fig. 3. Comparison Results Page

The following is the function of the implementation of the KMP and Boyer Moore algorithms using the PHP programming language.

KMP Function Algorithm Code

```
<?php
 function KMPSearch($p,$t){
   $hasil = array();
   $pattern = str_split($p);
   $text    = str_split($t);
   $lompat = $this->preKMP($pattern);
   $i = $j = 0;
   $num=0;
   while($j<count($text)){
     while($i>-1&&
      pattern[$i]!=$text[$j]){
        to the jump table
      $i = $lompat[$i];
     }
     $i++;  $j++;
     if($i>=count($pattern)){
       $hasil[$num++]=$j-count($pattern);
       $i = $lompat[$i];
     }
   }
 return $hasil;
  }

 function preKMP($pattern){
    $i = 0;
    $j = $lompat[0] = -1;
    while($i<count($pattern)){
      while
      ($j>-1&&$pattern
        [$i]!=$pattern[$j]){
         $j = $lompat[$j];
      }
      $i++;
      $j++;
      if($pattern[$i]==$pattern[$j]){
        $lompat[$i]=$lompat[$j];
      }else{
        $lompat[$i]=$j;
      }
    }
    return $lompat;
  }

   function KMPReplace($str1,$str2,$text){
      $num = 0;
      $location=$this->KMPSearch($str1,$text);
      $t = '';
      $n = 0; $nn = 0;
      foreach($location as $in){
          $t.=substr($text,$n+$nn,$in-$n-$nn).$str2;
          $nn = strlen($str1);
          $n = $in;
       }
    $t .= substr($text,$n+$nn);
     return $t;
  }
  ?>
```

BM Function Algorithm Code

```
<?php
function BoyerMoore($text, $pattern) {
 $patlen = strlen($pattern);
 $textlen = strlen($text);
 $table = makeCharTable($pattern);
 for ($i=$patlen-1; $i < $textlen;) {
     $t = $i;
   for($j=$patlen-1; $pattern[$j]==$text[$i];     $j--,$i--) {
          if($j == 0) return $i;
   }
   $i = $t;
   if(array_key_exists($text[$i], $table))
      $i = $i + max($table[$text[$i]], 1);
      else
      $i += $patlen;
   }
   return -1;
}

function makeCharTable($string) {
   $len = strlen($string);
   $table = array();
   for ($i=0; $i < $len; $i++) {
      $table[$string[$i]] = $len - $i - 1;
   }
   return $table;
}
```

C. Testing the KMP and BM Algorithms

The test results on 5 computer terms with the KMP and BM algorithm in table 3.

Table 3. the Test Results of a Computer Term

Algorithm	Process	Term of Computer	Memory usage	Time Needed
KMP	1	Abend	141208	0,0256
	2	Access	141208	0,0256
	3	Accumulator	141224	0,0256
	4	Balance Error	141224	0,0256
	5	Cache	141208	0,0256
BM	1	Abend	138136	0,0156
	2	Access	138136	0,0156
	3	Accumulator	138152	0,0156
	4	Balance Error	138152	0,0156
	5	Cache	138136	0,0156

ECM calculation results on the KMP and BM algorithm in table 4

Table 4. The Result of MPE Calculation

Process	Memory usage (Byte)			Time Needed (s)			Total	
	Weight	*KMP*	*BM*	*Weight*	*KMP*	*BM*	*KMP Value*	*BM Value*
1	0,5	141208	138136	0,5	0,0256	0,0156	375,94	371,79
2	0,5	141208	138136	0,5	0,0256	0,0156	375,94	371,79
3	0,5	141224	138152	0,5	0,0256	0,0156	375,96	371,81
4	0,5	141224	138152	0,5	0,0256	0,0156	375,96	371,81
5	0,5	141208	138136	0,5	0,0256	0,0156	375,94	371,79
Total		**706072**	**690712**		**0,128**	**0,078**	**1879,74**	**1858,99**

Total of value (ECM) = ΣN^{Weight}

The process of calculating the total value in the 1st process:

KMP value = $(141208 \text{ Byte})^{0,5} + (0,0256 \text{ s})^{0,5}$
= 375,78 + 0,16 = 375,94

BM value = $(138136 \text{ Byte})^{0,5} + (0,0156 \text{ s})^{0,5}$
= 371,67 + 0,12 = 371,79

The results of the comparison of the KMP and BM algorithm with MPE obtained time of search for BM algorithm is 37,9% and KMP algorithm is 62,1% while the memory usage of search for KMP algorithm is 50,6% and BM algorithm is 49,4%.

IV. Conclusion

Based on the total score obtained using the Experimental Comparison Method that the BM algorithm is 0.55% better than the KMP algorithm. The hypothesis of searching for word would be better to use the KMP algorithm turned out to be incorrect and the use of the KMP and BM algorithm in string matching has not too significant different advantages. But in developing a dictionary application search for computer terms recommended using the BM algorithm because it is faster in search process and more efficient in memory usage.

References

[1] R. Janani and S. Vijayarani, "An Efficient Text Pattern Matching Algorithm for Retrieving Information from Desktop," Indian J. Sci. Technol., vol. 9, no. 43, Nov. 2016, doi: 10.17485/ijst/2016/v9i43/95454.

[2] M. B. Sri, R. Bhavsar, and P. Narooka, "String Matching Algorithms," Int. J. Eng. Comput. Sci., vol. 7, no. 3, 2018.

[3] Diptarama, R. Yoshinaka, and A. Shinohara, "Fast Full Permuted Pattern Matching Algorithms on Multi-track Strings," in Proceedings of the Prague Stringology Conference 2016, 2016, pp. 7–21.

[4] P. Chettri and C. Kar, "Comparative Study between Various Pattern Matching Algorithms," in International Conference on Computing & Communication, 2016.

[5] H. Handrizal, A. Budiman, and D. R. Ardani, "Implementation and Analysis Zhu-Takaoka Algorithm and Knuth-Morris-Pratt Algorithm for Dictionary of Computer Application Based on Android," IJISTECH (International J. Inf. Syst. Technol., vol. 1, no. 1, p. 8, Nov. 2017, doi: 10.30645/ijistech.v1i1.2.

[6] R. Muntazari, A. Arini, and H. B. Suseno, "Application Of The Boyer Moore Method In The Application Dictionary Of Web-Based Information Technology Terms (Case Study: Pt. Erefka Tiga Pilar Utama)," Integr. (Journal Inf. Technol. Vocat. Educ., vol. 1, no. 2, 2019.

[7] S. Maryana, E. Kurnia, and A. Ruyani, "Web-based application on employee performance assessment using exponential comparison method," in IOP Conference Series: Materials Science and Engineering, 2019.

[8] Kholil, D. Prinajati, and N. A. Annisari, "Flood Management Model in Digital Era, Using SAST (Strtategic Assumption Surfacing and Testing) and the Exponential Comparison Method (ECM): A Case Study in Jakarta," J. Sci. Res. Reports, pp. 1–9, Aug. 2019, doi: 10.9734/jsrr/2019/v24i330156.

[9] C. S. Rao and S. V. Raju, "A Novel Multi Pattern String Matching Algorithm with While Shift," in Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies - ICTCS '16, 2016, pp. 1–5, doi: 10.1145/2905055.2905108.

[10] M. R. Apriyadi, E. Ermatita, R. F. Malik, and D. P. Rini, "Knuth Morris Pratt - Boyer Moore Hybrid Algorithm for Knowledge Management System Model on Competence Employee in Petrochemical Company," in 2019 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS), 2019, pp. 201–206, doi: 10.1109/ICIMCIS48181.2019.8985201.

[11] N. Kumari, "Online Assessment Of Similarity Between Sentences In Question Analogue Systems," University Phagwara, Punjab (India), 2016.

[12] R. Rahim, I. Zulkarnain, and H. Jaya, "A review: search visualization with Knuth Morris Pratt algorithm," in IOP Conference Series: Materials Science and Engineering, 2017.

[13] R. Rahim, A. S. Ahmar, A. P. Ardyanti, and D. Nofriansyah, "Visual Approach of Searching Process using Boyer-Moore Algorithm," in Journal of Physics: Conference Series, 2017.

[14] F. A. S. Cipriano, "Collaboration Application For Research Education Article Ranking, Using Social Tagging And Boyermoore Horspool String Pattern Algorithm," University of San Carlos Cebu City, Philippines, 2015.