An Integrated Supervised Learning Approach For High-Accuracy String Matching

M Musthafa Baig, PVRD Prasada Rao

Department of Computer Science & Engineering Koneru Lakshmaiah Education Foundation (KLEF), Green Fields, Vaddeswaram, A.P.- 522302. India.
**pvrdprasad@kluniversity.in*

Abstract
A fundamental problem in computer science is the string matching problem, which is the challenge of locating all instances of one string as a substring of another. Due to several applications in computational biology, this subject has recently got a lot of attention. Different ciphers are considered to speed up the search process in this research, which is a revised form of Horspool's string detection algorithm. The numerous pattern identification algorithms are used to locate all instances of a restricted set of patterns inside an input text or input file in order to examine the information of the documents. String matching can be done in one of two ways: exact matching or approximate matching. The proposed research focuses on employing an exact string matching using Inclusive Supervised Learning Model to develop a Accurate String Matching (ISL-ASM) that is an upgraded form of the Boyer-Moore-Horspool algorithm. When compared to traditional models, the proposed model's string matching accuracy is superior.

Keywords: String Matching, Boyer-Moore-Horspool Algorithm, Exact Matching, Variant, Supervised Learning, Input Query.

1. Introduction

String Matching can be used to solve problems in a variety of domains. Text mining and natural language processing, as well as image processing and speech processing, can all benefit from this technique. Natural language processing is required for multimedia information retrieval [1]. At the moment, most information retrieval research focuses on detecting and reading text in a variety of media, such as video or photographs. After retrieval methods recognize text with an optical character recognizer, string matching algorithms [2] are employed to find relevant phrases in the database. For example, the methods use picture or video tagging to find relevant phrases in multimedia databases for digitized texts, such as photos or videos with tags [3].

Query-based algorithms utilizing a string similarity measure were used to sift through the text. Recovery of multimedia objects such as text, images [7], and music is becoming increasingly popular as technology progresses [8]. The string matching process is represented in Figure 1.

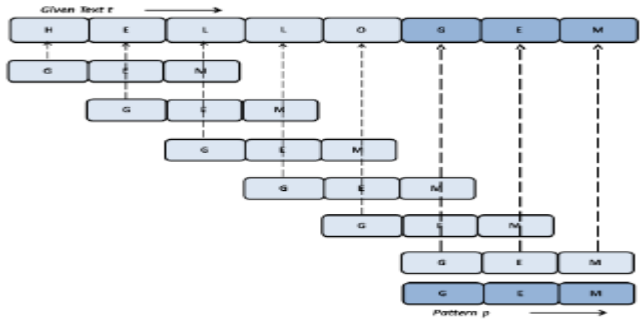

Fig 1: String Matching Process

It's common to see string matching utilized in a wide range of computer science applications. Information retrieval has always had this as a major concern [9]. A few standard methods have been implemented to the pattern recognition of text files, such as in data retrieval platforms and the editing of text. This character's methods serve as a model in disciplines of computer science system design, web search engines, malware signature matching, and networking [10]. They constitute the basis of the software implementation.

When using the Boyer-Moore algorithm, both bad character and excellent suffix strategies are used. This is the foundation of the algorithm [16]. A non-overlapping text file partition is used, and an overlapping text file partition is used. Each time, the remote server will be launched using the JAVA RMI method on a hypercube connected model to reduce search time [17].

A substring of a sentence appears in the matched string. To put it another way, text T={t1,t2.....tn} will be matched with pattern P={p1,p2.....pm}. One thing to keep in mind is that the text and pattern lengths must equal. Database schema, network systems, and other such systems use the matching string notion in real life [18]. The two strategies of precise and approximate matching are used to match occurrences of strings, respectively. The various types of string matching models are shown in Figure 2.

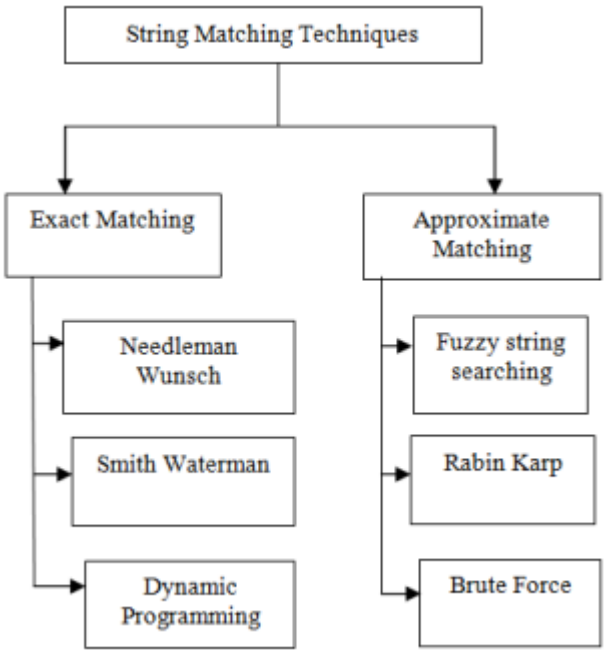

Fig 2: String Matching Technique

Algorithms that find a position inside a larger text or string where one or more strings might be identified are indeed an essential subclass of string algorithms. According to formal definitions, the pattern and the sought-after text are both's elements' vectors [19]. A typical human language could be the case. The text is taken to be an array T[1..n] with a specified length, and the pattern is taken to be an ensembles of length P[1..m] with a variable length, where m is equal to n's length [20]. Character matrices T and P are used to represent strings of characters in a programme. Numerous algorithms exist to deal with string matching. Single design and multi design algorithms are the two types of algorithms that can be divided based on the number of designs they search for. String matching can be accurate or imprecise depending on the requirements. The proposed research focuses on employing an exact string matching using Inclusive Supervised Learning Model [21] to develop a Precise String Matching Algorithm that is an upgraded form of the Boyer-Moore-Horspool algorithm that is used for accurate string matching with less time complexity.

2. Literature Review

Xiao Zhao et al.[1] tried to find a sequence match using a filtering technique with exact search portioning, neighborhood generation and Intermediate Partitioning (IP). It's also important to talk about the compression modules. These modules are going to teach all about pattern comparison, from generation to exact match and verification. Raita et al. [2] proposed the Raita technique as an alternative to the BM approach, which makes use of the strong interdependencies that arise between successive characters in a sentence. A symbol's dependencies can be extended up to 30 times. When applying this approach, the final letter of the sequence is compared to the text character on the right-hand side of the window after each try. To see if two patterns match, they are compared side by side. It is then determined if the two are compatible by comparing them side by side. If they don't, they're put up against each other to see if their characteristics match.

Navarro et al. [14] were able to improve on their previous findings by dividing the pattern in half. The new algorithm divides the input pattern in half, just like the old one did. Whereas in this work, the search process begins following the division into two parts, the pattern has been found if the distance between the patterns is zero; else, the search operation must be carried out again. A brute-force approach may take a long time to decode non-unicode texts, regardless of the fact that the recommended strategy shows promise in that regard.

3. Proposed Model

To see if the pattern's elements are present in an observed string sequence, use clustering. With the exception of analytical thinking, winning a match almost always requires pinpoint accuracy. If you want to match patterns, it is necessary to find the pattern's location in the string sequence that make a component of it, and replace the matching design with another text sequence. Sequences are frequently used to represent the patterns. The concept of pattern matching is widely used in a variety of fields.

The Boyer-Moore approach looks for patterns and text going right to left by starting with an integer equal to the length of pattern-1. Rather than matching the pattern's top, it resembles the pattern's bottom instead. When the alphabet and pattern are large enough, the best results can be achieved in the shortest amount of time. In determining the shifting index mapping, one uses a shift mapping or function that looks for matching characters or identifies a mismatch in the algorithm. The String matching procedure are represented in Figure 3.

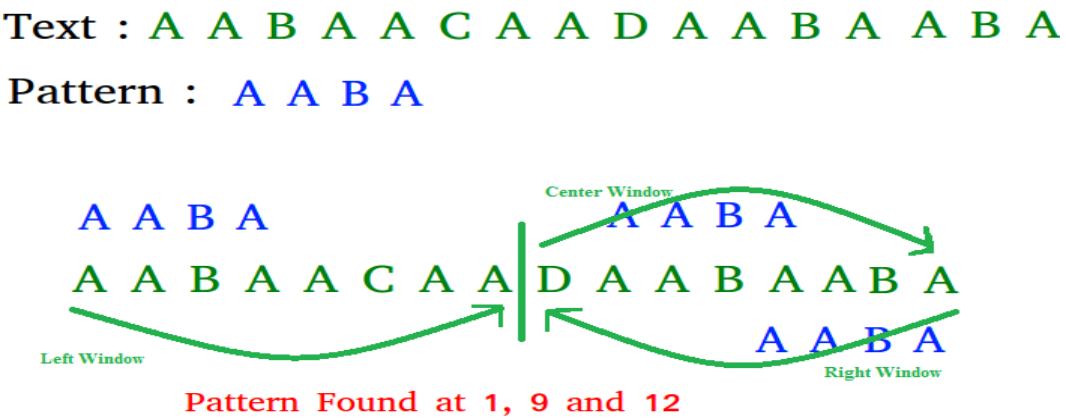

Fig 3: String Matching Procedure

The algorithm's syntax uses two functions to calculate the good suffix: the primary function and a function that determines the last occurrence. The algorithm keeps track of all the patterns and characters from the rightmost to the leftmost character. When there is a deviation or a complete match in a pattern, two pre-computed functions move the window to the right.

The primary disadvantage of Boyer-Moore type technique is the amount of time and space required for preprocessing, which is dependent on the size of the alphabet and/or the size of the pattern. As a result, if the sequence is short, it is preferable to apply the naive technique. To overcome the drawbacks of BM

algorithm, this research work proposes an Inclusive Supervised Learning Model to develop an Accurate String Matching Algorithm(ISL-ASM) that is an upgraded form of the Boyer-Moore-Horspool algorithm. The proposed model workflow is presented in Figure 4.

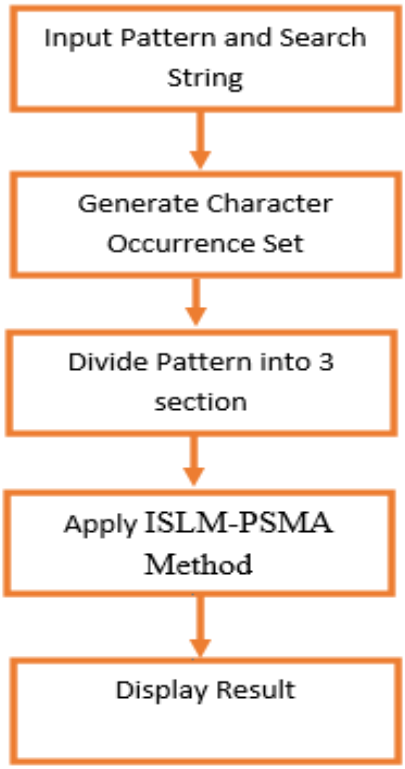

Fig 4: Proposed Model Framework

The proposed algorithm searches for the string totally from 3 sections of the pattern i,e from right, left and middle of the sequence towards right. The procedure is best suited for short size or long size patterns and the time complexity levels are also reduced and the accuracy levels are improved. The algorithm for ISL-ASM is presented here.

Algorithm ISLM-PSMA

{

Step 1: Initialize the Pattern and the Query-String as input.

Step-2: Initially identify each character count and its position in the provided pattern and store in a separate set.

$$\mathbf{Str\_Occ[M]=\sum i=0M\sum j=0N[len(String)compare(getChar(Str(i)),getChar(Str+i))]}$$

Step 3: Initialize pre-processing phase in which the text is divided into three sections, after which pre-processing for the pattern is carried out. The numbers in the left window set are calculated using the boyer-moore method, which includes two functions like bad character, good suffix rule that remain constant throughout the process. The shift values required to search for text on the left side of the pattern are maintained in the window set. The pattern is divided into 2 partitions and the central character is considered towards right of the sequence.

$$\mathbf{Right_{window}(Pattern(S))=\{i\in[M-1,0]|Pattern(S(M))=Character_{[i]}\}}$$

$$\mathbf{Left_{window}(Pattern(S))=\{i\in[0,M-1]|Pattern(S(i))=Character_{[i]}\}}$$

$Middle_{window}$(Pattern(S)={i∈[0,(M-1)/2]| Pattern(S((M-1)/2))=$Character_{[i]}$}

Step 4: Perform Pre-processing of the pattern considered in the multi directions for identification of character locations. The function for preprocessing of pattern is

Function Pre-Processing(String pattern)

```
Set ← Sizeof(pattern)
for i from 0 to (Sizeof(pattern)/2)
  do
      LSet[i] ← (Sizeof(pattern)/2)
  done
for i from (Sizeof(pattern)/2) to Sizeof(pattern)
  do
      RLSet[i] ← Sizeof(pattern)
  done
for i from Sizeof(pattern) to (Sizeof(pattern)/2)
  do
      RLSet[i] ← Sizeof(pattern)
  done
return Set
```

Step 5: Perform Similarity check of the Query_String and the pattern using the function Similarity_Check as represented

Iterative_Set-1:$Sim\_Check_{pattern}$ (Pattern,Query_String)={X,Y∈[0,Sizeof Pattern)/2]}

Iterative_Set-2: $Sim\_Check_{pattern}$ (Pattern, Query_String)={X,Y ∈[Sizeof(pattern)/2, sizeof(pattern)]}

Iterative_Set-3:$Sim\_Check_{pattern}$ (Pattern,Query_String)={X,Y∈ [Sizeof(pattern),sizeof(pattern)/2]}

Function Similarity_Check **(Pattern, Qurery_String, Sizeof(Pattern))**

```
X ← Sizeof(Pattern)- 1
Y ← Sizeof(Query_String)- 1
while Pattern[X[i]] = = Query_String[Y[i]]
   if status = = 0
      return "Matched"
      i ← i ++;
   else if  status > 0
```

return “Not matched”

$$\text{Sim\_Check pattern} := \begin{cases} \text{Matched} & \text{if status(pattern)} = 0 \\ \text{Not Matched} & \text{otherwise} \end{cases}$$

Step-7: After similarity is checked, the pattern matching is performed as per the ISLM-PSMA Model.

Function Pattern_Check(String pattern, String str)

{

$L_i \leftarrow 0$ to (Sizeof(pattern)/2)

i ← 0

Sim[]← N

F← 0

ITERATIVE SET-1:

```
for i from 0 to (Sizeof(pattern)/2)
  for j from 0 to Sizeof(str)
      if pattern[i] = = str[j]
            Sim[i]=str[j]
             i← i+1
             j ← j+1
            F++
       else
             j++
  if (F= = Sizeof(str))
     return “Matched”
  else
     return “Not Matched”
```

ITERATIVE SET-2:

```
for i from Sizeof(pattern)  to (Sizeof(pattern)/2)
  for j from Sizeof(str) to 0
      if pattern[i] = = str[j]
            Sim[i]=str[j]
             i← i+1
             j ← j+1
```

```
                F++
        else
                j++
    if (F= = Sizeof(str))
       return “Matched”
    else
       return “Not Matched”
```

ITERATIVE SET-3:

```
  for i from (Sizeof(pattern)/2)  to(Sizeof(pattern)
    for j from 0 to Sizeof(str)
        if pattern[i] = = str[j]
                Sim[i]=str[j]
                i← i+1
                j ← j+1
                F++
        else
                j++
    if (F= = Sizeof(str))
       return “Matched”
    else
       return “Not Matched”
```

Step-7: Display the Prediction result

The ISL-ASM algorithm is an advanced approach to string pattern matching, designed to optimize both preprocessing and comparison phases. Initially, it preprocesses the pattern by identifying character counts and positions, which takes linear time relative to the pattern length (O(M)). The algorithm employs the Boyer-Moore method during preprocessing, ensuring efficiency with a time complexity of O(M). The key innovation of ISL-ASM lies in its multi-directional comparison technique, which divides the pattern into sections and performs iterative matching against the query string. This process involves nested loops, resulting in a time complexity of O(MN). By using these sophisticated methods, ISL-ASM demonstrates superior performance compared to traditional single-directional algorithms, like the naive approach, which also has a time complexity of O(MN) but lacks the benefits of advanced preprocessing and segmentation.

Time Complexity Calculation:

1. **Preprocessing Phase:**
    - Identifying character counts and positions in the pattern: O(M)

- Boyer-Moore preprocessing: O(M)
- Combined preprocessing complexity: O(M)

2. **Pattern Matching Phase:**
   - Iterative comparisons of pattern sections with the query string:
     - Iterative Set 1: O(M/2×N)=O(MN)
     - Iterative Set 2: O(M/2×N)=O(MN)
     - Iterative Set 3: O(M×N)
   - Combined pattern matching complexity: O(MN)

When these phases are combined, the overall time complexity of the ISLM-PSMA algorithm is O(MN), highlighting its efficiency in handling extensive pattern matching tasks.

4. Results

There are two types of string matching algorithms: exact match and approximate match. When using exact pattern matching, the pattern must match exactly with the input text in order to display the beginning index position. The proposed model is implemented in python and executed using GOOGLE COLAB. The dataset link for string matching is available on https://www.kaggle.com/neelshah18/arxivdataset. The proposed Inclusive Supervised Learning Model to develop a Precise String Matching Algorithm (ISL-ASM) that is an upgraded form of the Boyer-Moore-Horspool algorithm is compared with existing compressed string matching algorithm based on run-length encoding (CSM-RL) Model. The proposed model is compared with the existing ones in terms of Pre Processing Time Levels, Data Clustering Time Levels, Clustering Accuracy, Precise String Matching Accuracy Levels, Precise String Matching Time Levels and False Positive Rate Levels.

Data validation and imputation are both part of preprocessing. Data preparation is crucial in machine learning processes to ensure that huge datasets are prepared so that learning algorithms can read and analyze the data they contain. Cleaning, instance selection, normalization, one-hot encoding, transformation, and feature extraction and selection are examples of data preparation. The results of final information processing can be perceived in different ways depending on how data was preprocessed. The pre-processing time levels of the proposed and traditional models are represented in Figure 5.

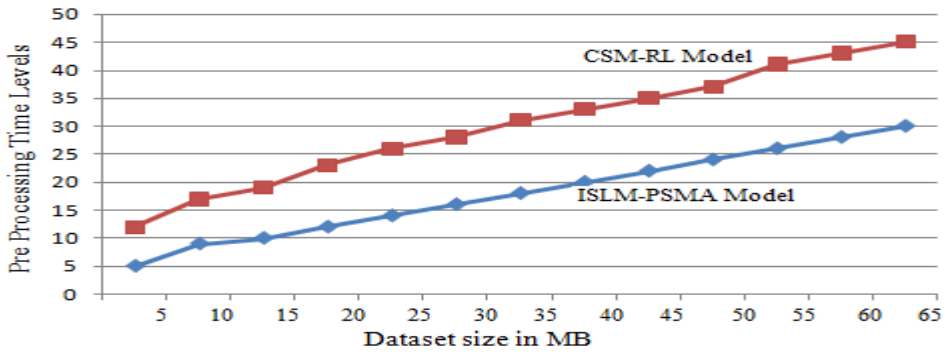

Fig 5: Pre Processing Time Levels

By classifying data into patterns that can be comprehended and manipulated more readily, clustering is commonly utilized. Clustering is the process of putting comparable data together into a single cluster and putting dissimilar data in separate clusters. It's called data clustering because it gathers related text data into a single collection. The proposed model completes the clustering process in a less time. The data clustering time levels of the proposed and existing models are represented in Figure 6.

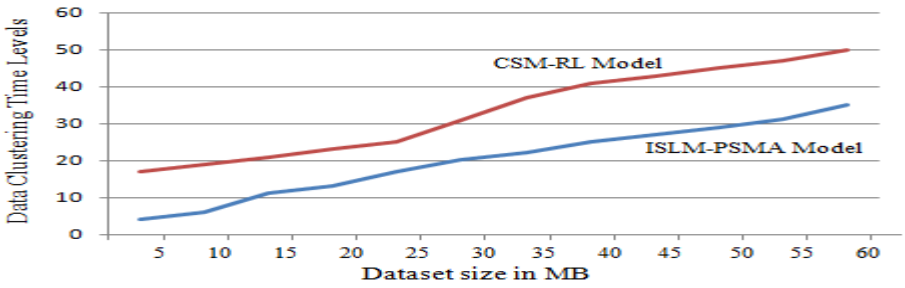

Fig 6: Data Clustering Time Levels

Data analysis and mining applications require a lot of clustering. In order to make things in one group more similar to one another, it's necessary to organize them into smaller groups. The clustering accuracy levels of the proposed model is high when compared to existing model. The accuracy level results are indicated in Figure 7.

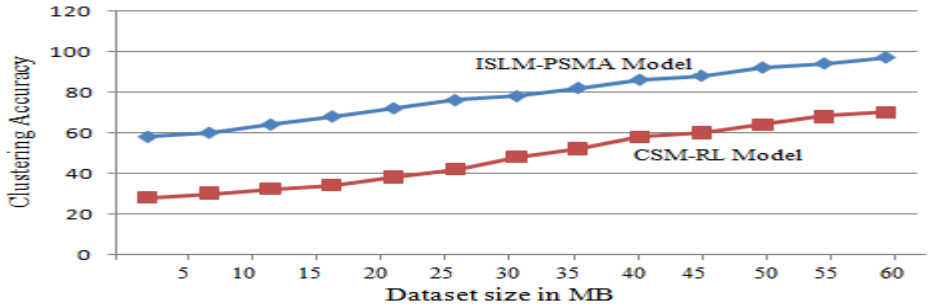

Fig 7: Clustering Accuracy

The problem of locating instances of a pattern string in a different string or text is called string matching. Searching for certain words or phrases is another term for this. Algorithms that match strings play a critical role in a wide range of real-world problems and applications. A few of its critical uses include Spell Checkers, Web Filtering, Intrusion Detection Systems, Internet Sites and Bioinformatics. The proposed model string matching accuracy is very high when compared to traditional method. The string matching accuracy levels of the proposed and traditional models are represented in Figure 8.

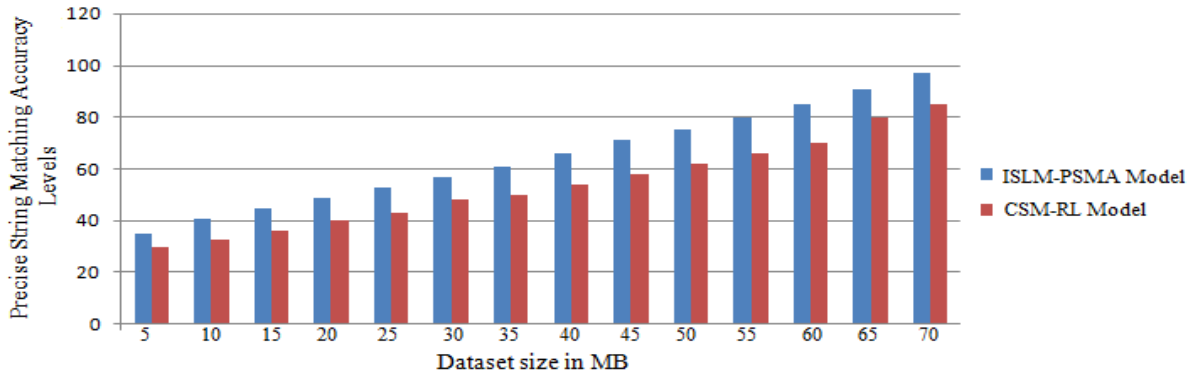

Fig 8: Precise String Matching Accuracy Levels

The proposed model takes less time in identification of a string from given text. The proposed model performs deep text analysis for the text recognition. The proposed model and traditional model string matching time levels are presented in Figure 9.

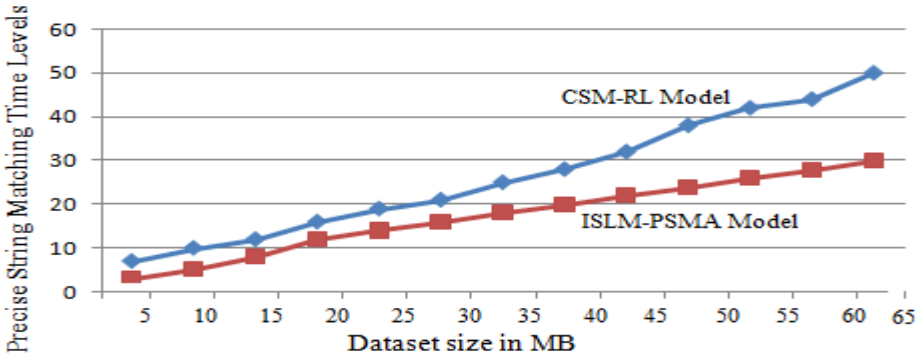

Fig 9: Precise String Matching Time Levels

When it comes to string matching accuracy, machine learning models tests the false positive rate (FPR) measures how accurate the model is performing. The proposed and traditional model false positive rates are shown in Figure 10.

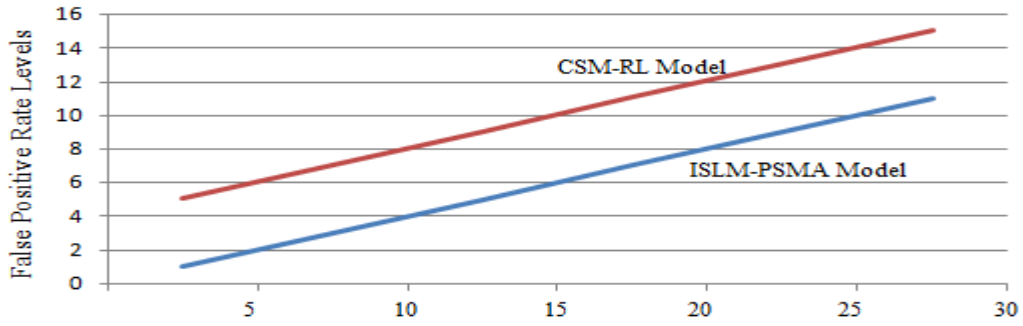

Fig 10: False Positive Rate Levels

5. Conclusion

Due to the growth of multiple algorithms, the field of pattern matching is extensive, and researching the technique, complexity, and constraints of all those methodologies is a time-consuming process. The proposed research focuses on employing an exact string matching using Inclusive Supervised Learning Model to develop a Precise String Matching Algorithm that is an upgraded form of the Boyer-Moore-Horspool algorithm. The proposed string matching algorithm is evaluated in each category based on time complication, limitation, and databases as a result of the analysis. We also highlighted new difficulties, applications, and directions for string matching algorithms to increase their scope. The research study is built on semantic-based approach concepts. It's very useful for retrieving multilingual data. In the proposed work, three different techniques are used to scan the search terms in the corpus, and the execution times for smaller and longer patterns are compared. By analyzing these graphs of algorithms, it can be indicated that the proposed model is efficient for smaller strings as well of large size text documents also. According to the research findings, the proposed algorithm provides the best accuracy for all inputs. In future, the research can be extended on character centric approaches for string matching to reduce the time complexity levels.

References

[1]. Xiao Zhao. An efficient pattern string matching algorithm [J]. Journal of Shanxi University of Science and Technology, 2017, 35(1):183- 187.

[2]. T. Raita, "Tuning the boyer-moore-horspool string searching algorithm,"Software: Practice and Experience, pp. 879-884, 1992.

[3]. M. Crochemore, Czumaj, A., Gasieniec, L., Jarominek, S., Lecroq, T.,Plandowski, W., Rytter, W., "Speeding up two string-matching algo-rithms," Algorithmica, pp. 247-267, 1994.

[4]. T. Berry, Ravindran, S., "A Fast String Matching Algorithm and Experi-mental Results," Stringology, pp. 16-28, 1999.

[5]. Xiao Zhao etc. An efficient pattern string matching algorithm [J]. Journal of Shanxi University of Science & Technology, 2017,35(1):183-187.

[6]. M. K. Ahmad, "An Enhanced Boyer-Moore Algorithm (Doctoral disserta-tion)," Middle East University, 2014.

[7]. D. M. Sunday, " A very fast substring search algorithm," Communications of the ACM,, pp. 132-142, 1990.

[8]. L. Colussi, "Correctness and efficiency of pattern matching algorithms,"(in English), Information and Computation, vol. 95, no. 2, pp. 225-251,Dec 1991.

[9]. H. Xian-feng, Y. Yu-bao, and L. Xia. "Hybrid pattern-matching algorithm based on BM-KMP algorithm." In Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on, vol. 5, pp.V5-310. IEEE, 2010.

[10]. Z. Cao, Y. Zhenzhen, and L. Lihua. "A fast string matching algorithmbased on lowlight characters in the pattern." In Advanced ComputationalIntelligence (ICACI), 2015 Seventh International Conference on, pp. 179-182. IEEE, 2015.

[11]. S. Hakak, A. Kamsin, P. Shivakumara, M. Y. Idna Idris, and G. A. Gilkar."A new split based searching for exact pattern matching for natural texts."PloS one 13, no. 7 (2018): e0200912. Skid

[12]. S. Hakak, K. Amirrudin, P. Shivakumara, and M. Y. I. Idris. "Partition-Based Pattern Matching Approach for Efficient Retrieval Of Arabic Text."Malaysian Journal of Computer Science 31, no. 3 (2018): 200-209.

[13]. R. M. Karp and M. O. Rabin, "Efficient Randomized Pattern-Matching Algorithms," (in English), Ibm Journal of Research and Development, vol.31, no. 2, pp. 249-260, Mar 1987.

[14]. G. Navarro, ‘‘Nrgrep: A fast and flexible pattern matching tool,’’ Dept. Comput. Sci., Univ. Chile, Santiago, Chile, Tech. Rep. TR/DCC-2000-3, Aug. 2000. [Online]. Available: ftp://ftp.dcc.uchile.cl/pub/users/gnavarro/ nrgrep.ps.gz

[15]. H.-T. Lu and W. Yang, ‘‘A simple tree pattern-matching algorithm,’’ in Proc. Workshop Algorithms Theory Comput., 2000, pp. 1–8.

[16]. Y. Bai and H. Kobayashi, ‘‘New string matching technology for network security,’’ in Proc. 17th Int. Conf. Adv. Inf. Netw. Appl. (AINA), Mar. 2003, pp. 27–29.

[17]. M. Ahmed, M. Kaykobad, and R. A. Chowdhury, ‘‘A new string matching algorithm,’’ Int. J. Comput. Math., vol. 80, no. 7, pp. 825–834, Jul. 2003.

[18]. D. Cantone and S. Faro, ‘‘Fast-search: A new efficient variant of the Boyer-Moore string matching algorithm,’’ in Experimental and Efficient Algorithms (Lecture Notes in Computer Science), vol. 2647, K. Jansen, M. Margraf, M. Mastrolilli, and J. D. P. Rolim, Eds. Berlin, Germany: Springer, 2003.

[19]. D. Cntone and S. Faro, ‘‘Searching for a substring with constant extra-space complexity,’’ in Proc. 3rd Int. Conf. Fun Algorithms, 2004, pp. 118–131.

[20]. G. Navarro and M. Raffinot, ‘‘A bit-parallel approach to suffix automata: Fast extended string matching,’’ in Combinatorial Pattern Matching (Lecture Notes in Computer Science), vol. 1448, M. Farach-Colton, Ed. Berlin, Germany: Springer, 1998.

[21]. B. U. J. Holub, ‘‘Fast variants of bit parallel approach to suffix automata,’’ in Proc. 2nd Haifa Annu. Int. Stringol. Res. Workshop Israeli Sci. Found., 2005, pp. 1–20.

[22]. M. E. Nebel, ‘‘Fast string matching by using probabilities: On an optimal mismatch variant of Horspool’s algorithm,’’ Theor. Comput. Sci., vol. 359, nos. 1–3, pp. 329–343, Aug. 2006.

[23]. Huiming He etc. A multiple pattern string matching algorithm based on substring recognition [J]. Computer Applications and Software, 2011, 28(11):10-14.

[24]. Wenzhi Li etc. A string matching technique for large-scale collection of short feature [J]. Computer Engineering and Applications,2014,50(1): 105-110.

[25]. Shibo Dong etc. An improved string pattern qppmatching algorithm [J]. Computer Engineering and Applications, 2013,49(8):133-137.

[26]. Shengdong Dai etc. A matching algorithm for computing pattern features of DNA sequence [J]. Journal of Electronic University of Science &Technology of Hangzhou,2015,35(1):88-92.

[27]. Gaochao Sun etc. Method for improving pattern matching performance by using parallel computing[J]. Journal of Information Engineering University, 2011, 12(6):750-753.

[28]. Yi Zhuliang On Improved Bm Pattern Matching Algorithm Based On Network Intrusion Detection System[J]. Computer Application And Software, 2012, 29(11) :193- 196.

[29]. Wang H, Zhang L. Quick Pattern Matching Algorithm Based On Bad Character Sequence Checking[J]. Computer Applications And Software, 2012, 29(5) :114-116.

[30]. Yan H. Research On Improved Bmh Single-Pattern Matching Algorithm Based On Snort[J]. Computer Engineering And Applications, 2012, 48(31):78-81.