International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Efficient Substring Pattern Matching Technique for Hindi Language Text

Barkha Sahu

Submitted: 03/11/2022 **Revised:** 07/12/2022 **Accepted:** 15/12/2022

Abstract: With the exponential growth of internet users, the demand for regional language support, particularly Hindi, has significantly increased. However, the digital ecosystem remains predominantly optimized for the English language, resulting in a subpar experience for native Hindi users. Addressing this gap necessitates the development of efficient computational techniques tailored for Hindi language processing. One of the critical challenges in this domain is efficient string searching, which forms the basis for numerous applications such as text retrieval, data mining, and natural language processing. This research presents a novel substring searching algorithm specifically designed for the Hindi language. The proposed approach focuses on minimizing the number of character comparisons required to locate a pattern within a given string, thereby improving search efficiency and overall computational performance. By leveraging the unique characteristics of the Hindi script and optimizing the search mechanism accordingly, our algorithm enhances processing speed while maintaining high accuracy. This advancement not only contributes to the field of computational linguistics for Hindi but also promotes broader inclusion of regional languages in digital platforms, ensuring a more equitable and seamless user experience for the growing base of Hindi-speaking internet users.

Keywords: *Pattern, Substring Searching, Unicode ,Hindi Language Processing, String Searching, Substring Matching, Computational Linguistics, Pattern Detection.*

1. Introduction

A string is made up of numerous alphabets that have been concatenated together. These alphabets have a distinct number for each language. The process of finding the occurrence of a shorter string called Pattern in a much larger string called Text is said to be Substring Pattern Matching, which is used to solve a variety of real-time issues in computer science, including spam filtering, intrusion detection, DNA sequencing, and many more. Furthermore, Substring Pattern Matching assists in time-saving activities across a wide range of industries. The Algorithms for Substring Pattern matching may be divided into two categories. Exact String Matching, where exact pattern matching is necessary, such as Robin Karp, Boyer Moore, and KMP Algorithms, and Approximate String Matching, where fuzzy matching, i.e. approximate matching, is also permitted, such as Hamming Algorithms. Leven shtein edit distance and edit distance. There are many algorithms available for the English language. According to a Google-KPMG analysis, by 2021, an estimated 536 million Indians would utilize regional languages while online, aided by improving device affordability and data rates, as well as the availability of more local content. According to the research, by 2021, Hindi Internet users (201 million) would outnumber English Internet users (199 million). By the same period, India is predicted to have 735 million Internet users, up from 409 million in 2016. Thus there is a requirement for development of algorithms that can work efficiently over Hindi Language.

1Assistant Professor, Department of Computer Science & Engineering, Institute of Engineering & Science, IPS Academy, Indore, India.
Email Id: bsahuscs@gmail.com
Corresponding Author: Barkha Sahu
Email Id: bsahuscs@gmail.com

2. Related Work

With the rise in the No. of Hindi Users over the internet the requirement of better representation of Hindi Language over Computer system has increased a lot.Representation of Hindi Language dates back in 1990's where different attempts were made by developing compiler and text editor for Hindi Language [1].

Moreover In [2] Sahu and Joshi have taken the work further and done the statistical analysis of Hindi Language for creating a Hindi Dictionary.

Joshi and Kushwah [3] gave an error detection method specifically for Hindi Language when it is represented in a computer.

In [4] Natural Language processing over Hindi Language is applied for enhancing its usage in computer systems. From OCR, character recognition algorithms for Hindi have been described by Kushwah and Joshi [5].

In further work they have applied the NLP and Text recognition methods for detecting Hindi Language Poems for text script.[6]. More generalized work by Wissink [7] has demonstrated the use of unicode for representing Indic language which is essential for Globalized Software Requirement. A.Rathod et al [8] have described about the difficulties in typing Indian Languages and the resolution methods for Computer Representation of Indian Languages

taking telugu as the use case In [9] A.Joshi et al have designed a new layout of Keyboard for typing using the Indic alphabet structure. Dongre and Mankar [10] have described the database of handwritten numerals and characters of Devanagari Script and its usage in Hindi handwriting recognition in their Article. A time efficient model for Substring searching by taking various prefinded algorithms and brute force methods in consideration is described by Lumburovska [11]. From the related work observed in this section it is clear that a huge amount of work has been done for representing Hindi Language in Computers. Now performance enhancement is the crucial issue which is taken into account for this experiment.

3. Methodology

In this experiment as elaborated in the unicode official document, there are 128 different characters encoded in unicode for hindi language which contains hindi characters and hindi numerals too. Devanagari ranges from 0900 to 097F in hexadecimal coding. The Proposed BSBJ-Algorithm consists of two parts. The working of first part is presented in Figure-1 ,It preprocesses the string to make the master index list where the indices of each character are stored. And the functioning of second part is depicted in Figure 2, for the pattern searching. The algorithms are explained below :

Preprocessing

1. Start by taking the string in which the pattern is to be found in.
2. Initialize 128 different data variables may be array or list depending on the choice of implementational language, one for each Hindi character encoded in unicode.
3. Then one by one take the characters in the string and identify the unicode value of the character.
4. Now on the basis of the unicode enter the index value of the character in the specific variable defined in step 1
5. Repeat the process observed in step 3 and step 4 for all the characters in the string
6. Now we have a master index list storing the occurrence of each character in the string.

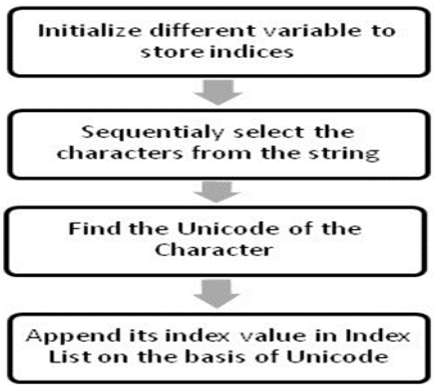

Figure 1 : Preprocessing Proposed Algorithm

Pattern Searching Algorithm

1. Take the pattern, the string, and the Master Index List
2. Initialize a count variable to store the number of Comparisons
3. Firstly calculate the length of the pattern and store it in a variable.
4. Then take the first character of the pattern, find its unicode and on the basis of the unicode. Identify which data variable stores its occurrence in the string. Also increase the count variable.
5. Take the variable where all the occurrences of the first character of the pattern are there as the index in the string.
6. Now one by one starting from the first index value to the last index value stored in the variable specified in step 4, repeat the below. given steps.
7. Starting from the index value to the pattern length, cut the sub string from the Main string and store it in a temporary variable.
8. Now match substring identified in step 6 with pattern. And the count variable is also increased. If a perfect match is found, print the index and the no. of comparison else shift to the next index value.

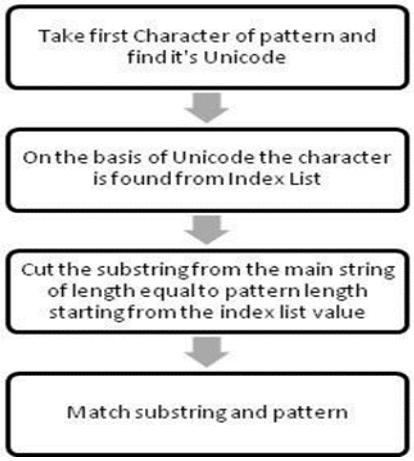

Figure 2 : Proposed pattern searching algorithm

In the next section we present the proposed BSBJ algorithm and observation from the execution of algorithm in Table 1.

4. BSBJ Algorithm

Initialisation Steps:

- ➔ Let <S>be the source string of length|S|
- ➔ Let <P> be the Pattern string of length|P|
- ➔ Items in source string are indexed by i.
- ➔ Items in pattern string are indexed by j.
- ➔ K[M][N] is a 2D array which acts as theMaster Table .

S.No.	Pattern	Index in the string	No. of Comparison
1	अकवारना	49	10
2	अंगसेवक	1883	270
3	अ त	8719	1176
4	कारोबार	75369	5078
5	कंु भ	78900	5685
6	यपण	202729	7548
7	बंदनी	216001	1882
8	सयू	38549	787
9	सेवा	21301	345

Preprocessing Steps:

- ➔ Step1:Start from i to i=s-1
- ➔ Step 2:Take S[i] and calculate its unicode value.
- ➔ Step 3: According to the unicode value append i in the Master Table where M = unicode character, N=index of i.
- ➔ Step 4: Stop

Pattern Matching Steps :

- ➔ Step 1:Take j= 0 for calculating unicode of P[j]
- ➔ Step 2: Select the row from the Master Tableon the basis of the unicode of P[j].
- ➔ Step 3: Let r be the no. of element in row, Nis used for index start N=0 to N=r-1.
- ➔ Step 4: K[M][N] where M= unicode valueand N= 0 so K[unicode][0]
  Starting from S[N] to S[N+|P|] Where, X be the intermediate pattern If x==p
  then
  Pattern found at n
  else
  N++

Repeat step 4,

- ➔ Step 5: Stop

5. Result

In this experiment, we have considered a dataset of 49,000 Hindi words. Firstly we have made a random string by randomly concatenating the wordsfrom the dataset. Then sequentially the words from the dataset are tested for the occurrence in the createdstring by using the proposed algorithm from the methodology. For sample, we have presented a table 1 where the pattern, its found index and no. of comparison to find it are stated. A plot of index in the string (Blue Line) and No. of Comparisons for finding the pattern in the string (Red Line) is presented in Figure 3. The key observation for table 1and figure 3 is that there exist no relation in index and No. of Comparison making this algorithm better than conventional algorithms.

Table 1 : Execution outcome as Sample Results

Table 2 contains statistical information found in the experiment.

10	हठ	132697	1398

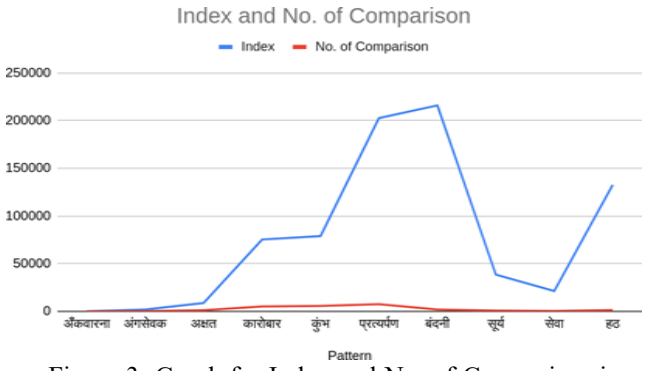

Figure 3: Graph for Index and No. of Comparison in observed pattern

Table 2 : Performance Value of Various Algorithms

Parameters	Proposed Algorithm
Minimum Comparison	2(अकँ ड़ी)
Maximum Comparison	19909 (रौहाल)
Average Comparison	4097.317
Standard Deviation	3576.936
Coefficient of Variation	0.872

6. Conclusion

From the experiment conducted and the observed results, it can be concluded that our algorithm has reduced the number. There is the least number of comparisons required to match a pattern in the string. But the solution of a process involves a process. Also, since our data is stored in Unicode UTF-8 format, the memory space required to store our database is very less.

Reference

[1] B.K. Joshi, 1998 "On the Design and Implementation of Computer Software and Software Development Tools in Hindi" Ph.D. Thesis, Rani Durgavati University, Jabalpur.

[2] B.Sahu, B.K.Joshi(2020) "A Tool for Statistical Analysis of Alphabets and Words of Hindi. In: Kumar A., Paprzycki M., Gunjan V. (eds). Lecture Notes in Electrical Engineering,vol 601.Springer, Singapore. ICDSMLA 2019 pp 1237-1244.

[3] B.K.Joshi and K.K.Kushwaha, November 2016, "Micro-Parsing of Hindi Words", International Journal of

Computer Science and Information Security", Vol-14, No 11, pp. 261-265.

[4] B.K.Joshi and K.K.Kushwaha, “Sandhi: The Rule-Based Word Formation In Hindi”, December 2016, International Journal of Computer Science and Information Security, Vol-14, No 12, pp. 781-785.

[5] K.K.Kushwaha and B.K.Joshi, Nov18-19, 2016, “Hindi Modifier Recognition based on Pixel Relationship”, International Conference on ICT Business Industry Government, Indore, Pp.1-6.

[6] K.K.Kushwaha and B.K. Joshi March 2017 “Rola An Equi-Matrik Chhand of Hindi Poems”, International Journal of Computer Science and Information Security(IJCSIS), Vol. 15, No. 3, pp.362-364.

[7] C. Wissink, “ Issues in Indic Language Collation ,” in 19th International Unicode Conference San Jose, 2001, pp. 1–9.

[8] A.Rathod, A.Joshi, U.Athavankar, “Text Input Methods For Indian Languages”, Master of Design Project Thesis, Industrial Design Centre, Indian Institute of Technology,Bombay, 2002.

[9] A. Joshi, A.Ganu, A.Chand, V. Parmar, G.Mathur , “Keylekh: a keyboard for text entry in indic scripts” Conference Paper · January 2004 DOI: 10.1145/985921.985950

[10] V.J. Dongre and V. H. Mankar, “Development of comprehensive devanagari numeral and character database for offline handwritten character recognition,” Applied Computational Intelligence and Soft Computing, vol. 2012,January 2012 Article No.: 29 pp. 1–5.

[11] Lina Lumburovska, “Time-Efficient String Matching Algorithms and the Brute-Force Method” Bachelor Thesis University Study Program First Cycle Computer And Information Science , 2018 Ljubljana.