International Journal of Advanced Technology in Engineering and Science
Vol. No.5, Issue No. 06, June 2017
www.ijates.com
ijates
ISSN 2348 - 7550

AN EXACT PATTERN MATCHING METHODOLOGY USING TEXT WINDOWS

Renuka S. Matkar[1], Yogeshwar. M. Kurwade[2], Dr. Vilas M. Thakare[3]

[1,2,3]*SGBAU, Amravati, Maharashtra (India)*

ABSTRACT

String pattern matching or searching is the act of checking for the presence of the constituents of a given pattern in a given text where the pattern and the text are strings over some alphabet. Pattern matching plays an important role in many computer related fields, such as information retrieval, intrusion detection, data compression, content filtering, gene sequence comparison and computer virus signature matching. Pattern matching algorithms belong to computationally intensive algorithms. String matching is further divided into two classes exact and approximate string matching. In exact String matching, pattern is fully compared with the selected text window (STW) of text string and display the starting index position. Improved-Bidirectional (IBD) exact pattern matching algorithm introduced a new idea of scanning partial text window (PTW) as well with the pattern to take decision of moving pattern to the right of partial text window. In this paper, an exact pattern matching methodology is used to retrieve the efficient text from an input query and then stop words are removed then the grammar checking is performed and then ungrammatical words are retrieved and pattern matching algorithm is applied to ungrammatical words and search in the relevant datasets for pattern matching.

Index Terms: Algorithm, exact pattern matching, pattern matching, Bidirectional, Improved Bidirectional, text window, window sliding.

I. INTRODUCTION

String matching algorithms or string searching algorithms are a dominant class of the string algorithms which aim to find one or all occurrences of the string within a larger group of the text [1]. String matching is further divided into two classes exact and approximate string matching. Exact pattern matching algorithms aims to find one or all occurrences of string within a larger group of text string [1]. In exact pattern matching, pattern is fully compared with selected text window of text string. In approximate string matching, if some portion of the pattern matched with STW and then it displays the results. The purpose of string matching algorithms is to find all occurrences of the pattern in the given text string [2]. Pattern matching algorithms belong to computationally intensive algorithms. It can be classified into single pattern matching and multi-pattern matching according to the number of matching patterns. Single-pattern matching came first, and there are some classic algorithms such as the Knuth-Morris-Pratt (KMP) algorithm and the Boyer-Moore (BM) algorithm, which offer some lessons and inspirations for the development of later multi-pattern matching algorithm. Multi pattern matching algorithm can find all occurrences of multiple patterns with only once scanning. It is more complex to implement but a wider range of applications than single-pattern matching algorithm. In a multiple pattern

matching, from a given finite set of strings (patterns), it can locate all patterns' locations simultaneously in a text string. Multiple pattern matching algorithms can be used in data mining area to find selected interesting patterns, security area to detect certain suspicious keywords or even biological field for DNA searching [3]. Pattern matching plays an important role in many computer related fields, such as information retrieval, intrusion detection, data compression, content filtering, gene sequence comparison and computer virus signature matching. It is a basic problem in computer science. Pattern matching is one of the major issues in the area of computational biology. Biologists search database for important information in different directions. Pattern matching will continue to grow and need changes from time to time. The analysis of Protein and DNA sequence data has been one of the most active research areas in the field of computational molecular biology [4]. The pattern matching algorithm is the core algorithm of the entire anti-virus software. Combining the advantages of fast calculation of hash function and parallel pattern matching of automata, it has significant performance advantages in the circumstance of virus signature matching [5].

In this paper, an exact pattern matching method using text windows is proposed which will be helpful in various needs of pattern matching and searching. Section 1 gives the brief introduction of the exact pattern matching methodology. This paper describes the basic concept and working of exact pattern matching methodology for efficient text retrieval. Then new algorithm is compared with some existing algorithms in terms of their comparison order, preprocessing space complexity, preprocessing time complexity and searching time complexity.

II. BACKGROUND

Along with the rapid development of computer technology, people's lives are increasingly dependent on computers. The rapid development of the Internet increases the freedom of application. But at the same time, because of its inherent openness, universality and freedom, it requires a higher demand of information security. Pattern matching plays an important role in many computer related fields, such as information retrieval, intrusion detection, data compression, content filtering, gene sequence comparison and computer virus signature matching. Basic need of pattern matching and applications are discussed in [1].

String matching is further divided into two classes exact and approximate string matching. Exact pattern matching algorithms aims to find one or all occurrences of string within a larger group of text string. The types of exact pattern matching and various problems and their solutions are discussed in [2].

In exact pattern matching, pattern is fully compared with selected text window of text string. In approximate string matching, if some portion of the pattern matched with STW and then it displays the results. The purpose of string matching algorithms is to find all occurrences of the pattern in the given text string. The working of previous existing pattern matching algorithm is proposed in [3].

Single-pattern matching appeared first, and there are some classic algorithm such as the Knuth-Morris-Pratt (KMP) algorithm and the Boyer-Moore (BM) algorithm, which offer some lessons and inspirations for the development of later multi-pattern matching algorithm. Aho-Corasick algorithm, a variant of the Knuth-Morris-Pratt algorithm, was the first algorithm to solve the multiple string pattern matching problems in linear time based on automata approach. Commentz-Walter presented an algorithm for the multi-pattern matching problem that combines the Boyer-Moore technique with the Aho-Corasick algorithm. Commentz-Walter combines the

filtering functions of the single pattern matching Boyer-Moore algorithm and a suffix automaton to search for the occurrence of multiple patterns in an input string. Wu-Manber algorithm is a simple variant of the Boyer-Moore algorithm that uses the bad character shift for multiple pattern matching. Mostly used algorithms for multiple pattern matching are discussed in [4].

Brute-force algorithm performs a checking, at all positions in the text between 0 and n-m, whether an occurrence of the pattern starts there or not. The Boyer-Moore Algorithm works with a backward approach, the target string is aligned with the start of the check string, and the last character of the target string is checked against the corresponding character in the check string. The Index based Pattern Matching using Multithreading method performs pre-processing to get the index of the first character of the pattern in the given text. Improvement in the existing algorithms is discussed in [5].

This paper is organized as follows, Section I gives the brief description about the pattern matching algorithms and it's applications along with proposed algorithm. Section II gives basics about the working and hierarchy of various pattern matching algorithms. This paper gives us an idea to improve the existing exact pattern matching algorithm with new advancements. This paper describes the concept and working of exact pattern matching methodology using text windows. Then new algorithm is compared with some existing algorithms in terms of their comparison order, preprocessing space complexity, preprocessing time complexity and searching time complexity. Then the results and discussions that compare the IBD algorithm with existing algorithms are presented. Finally, the conclusions from the experiments are drawn.

III. PREVIOUS WORK DONE

S.Nirmala Devi et al. (2012) [4] and Dr.S.P. Rajagopalan et al. (2012) [4] proposed the Index based Pattern Matching using Multithreading method performs pre-processing to get the index of the first character of the pattern in the given text. By using this index as the starting point of matching, it compares the Text contents from the defined point with the pattern contents. Raju Bhukya et al. (2012) [4] and DVLN Somayajulu et al. (2011) [4] proposed IBSPC. In IBSPC indexes has been used for the DNA sequence. After creating the index the algorithm will search for the pattern in the string using the index of least occurring character in the string.

Raju Bhukya et al. (2010) [3] and DVLN Somayajulu et al. (2010) [3] proposed IFBMPM. In IFBMPM to search some pattern P in text S, it start the search from the indexes stored in the row of index table which corresponds to the first character of the pattern P. If any character mismatches in its position, we skip the search and go for the next index which corresponds to the first character of the pattern P according to the indexes stored in index table for matching.

Reverse Colussi et.al. (2009) [1] proposed RC algorithm in which comparisons are done in specific order given by the preprocessed phase. The time complexity of preprocessing phase is O(m2) and searching phase is O(n). When several pattern strings need to match, using Single pattern matching has low efficiency. Corasick M.J et.al (2008) [1] proposed many pattern matching algorithm with high efficiency to solve this problem, which is called for short AC algorithm. In the preprocess stage, AC algorithm form several pattern strings waiting for matching, according to their features into Tree finite state automata, and decide the next situation according to matching characters.

Brute force *et.al* (1994) [2] proposed brute force or Naive algorithm which is the logical place to begin the review of exact string matching algorithms. It compares a given pattern with all substrings of the given text in any case of a complete match or a mismatch. It has no preprocessing phase and did not require extra space. The time complexity of the searching phase of brute force algorithm is o(mn).

Boyer-Moore *et.al*. (1977) [1] proposed Boyer-Moore (BM) algorithm published in 1977. It performed character comparisons in reverse order from right to the left of the pattern and did not require the whole pattern to be searched in case of a mismatch. The time and space complexity of preprocessing phase is O(m+|∑|) and the worst case running time of searching phase is O(nm + |∑|). The best case of Boyer-Moore algorithm is O(n/m).

Boyer-Moore Horspool *et.al.* (1990) [3] proposed BMH which did not use the shifting heuristics as Boyer-Moore algorithm used. It used only the occurrence heuristic to maximize the length of the shifts for text characters corresponding to right most character of the pattern. Its preprocessing time complexity is O(m+|∑|) and searching time complexity is O(mn). Authors proposed window sliding method.

Turbo Boyer Moore *et.al* (1994) [5] proposed TBM algorithm is variation of the Boyer-Moore algorithm, which remembers the substring of the text string which matched with suffix of pattern during last comparisons. It does not compare the matched substring again; it just compares the other characters of the pattern with text string.

IV. EXISTING METHODOLOGY

A. Bidirectional Exact Pattern Matching

In Bidirectional exact pattern matching algorithm, it compares a given pattern with selected text window (STW) from both sides, simultaneously, one character at a time within the text window [1]. It did not require the whole pattern to be searched if a mismatch occurs. In case of a mismatch or a complete match of the pattern, the mismatched and right pointers scan for the mismatched and rightmost characters of the STW to the left of the related text characters in pattern at same shift's length. Then align the pattern to new selected text window of string when rightmost and mismatched characters matched at same shifts in left of pattern. A complete match will be found when the both left and right pointers cross each other at the middle of the pattern. Bidirectional EPM algorithm is basically based on the bad character rule of Boyer-Moore algorithm where only one character is used to identify the shifts [1]. If the first mismatch is found at position i and T(k) is the text character that mismatches p(i) then shift the pattern right by following mathematical function.

$$max(1, i-R(T(K)))$$

Where, R(T(K)) is the mismatched character found in right shift.

B. Improved Bidirectional Exact Pattern Matching

Improved Bidirectional EPM algorithm scans partial text window of the text string for leftmost character of pattern and pattern for the rightmost character of the partial text window. Improved Bidirectional EPM algorithm compares a given pattern character wise with selected text window from both sides simultaneously as Bidirectional (BD) algorithm does. In this method, max shift rule is used for pattern matching [2].

$$max\ shift\ rule = max(d_i, d_{m-1})$$

C. A Two Way Pattern Matching Algorithm

An efficient pattern matching algorithm based on preprocessing of the pattern string by considering three consecutive characters of the text that immediately follow the aligned pattern window in an event of mismatch between pattern and text character. The algorithm makes use of two sliding patterns. It used the bad character of shift right where indices of three successive characters a, b, c from right are (n-m-3), (n-m2), (n-m-1) respectively are computed based on the mathematical formula shown below [3]:

$$Bad\ char\ of\ shift\ r(a,b,c) = min\begin{cases} m+2 \\ m+1 \\ m+3 \end{cases}$$

D. Naive Pattern matching algorithm

In this type, Pattern is set to the left end of the text and it stops when it found its match Brute force is another name for this type and if any mismatch is found pattern shifts to right one place. This algorithm needs no preprocessing. It used min shift rule in the preprocessing phase [4].

$$\min shift\ rule = \min(d_i, d_{m+1})$$

Where, d_i and d_{m+1} are the two shift decisions of the shift table respectively.

E. A2KD string pattern matching algorithm

It focuses on new algorithm which will consume less time as compared to present algorithms in the various application areas like intrusion detection, detecting plagiarism, bioinformatics, digital forensic and exploring text etc. In the proposed scheme, A2KD algorithm uses the different execution of a loop than existing. So, it takes much lesser execution time. The shift table d defined for each symbol a in alphabet Σ as shown below:

$$d(a) = mins|s = mor(1 \leq s < mandpm - s = a)$$

Where, d(a) is a shift table d. Mins is a minimum shift function.

V. ANALYSIS AND DISCUSSION

The worst case time complexity of preprocessing phase of Bidirectional EPM algorithm is *O(m)*, because only one loop is used to scan the pattern and partial text window to find the rightmost character of partial text window and left-most character of pattern [1].

In the Improved-Bidirectional EPM algorithm, as it is improved, inner while loop of searching phase of Improved-Bidirectional EPM algorithm runs at most *'m/2'* times so, its worst case time-complexity is *O(m/2). It is* because two pointers are used. The worst case time-complexity to shift pattern to right of the text is *O(n)* because the external while loop runs 'n' times at most. The total time complexity of searching phase is *O(m/2).O(n)*, it is because the inner loop runs within external while loop. It can be written as *O(mn/2).* Improved-Bidirectional EPM algorithm requires *O(m)* extra memory space in worst case to execute in addition with the pattern and text string [2].

There are two variables that run simultaneously in the program one variable is for the length of the string and other variable is for the length of the pattern that is to be searched in the string. During execution the main loop works until the pattern is being searched successfully once it is being found it stops it search further and displays the result. This is different and less time consuming as compared to other algorithms. In this algorithm the loop

runs less number of times that is the difference between the string lengths of the main string and the pattern which is to be found [3].

Naive pattern matching algorithm is the simplest among number of algorithms. In this, pattern is set to the left end of the text and it stops when it found its match Brute force is another name for this type and if any mismatch is found pattern shifts to right one place. This algorithm needs no preprocessing [4].

A2KD algorithm uses the different execution of a loop than existing. So, it takes much lesser execution time [5].

Table 1: Comparison of Improved-Bidirectional EPM with Existing Algorithms

Methodology/ Algorithm	**Advantages**	**Disadvantages**
1. Bidirectional exact pattern matching	1.Searches pattern from both sides 2. A complete match is found at end.	1. As both the sides are being searched, more time is used for searching 2. Suitable for smaller strings only
2. Improved Bidirectional Exact Pattern Matching	1. Efficient than the existing algorithms. 2. Improved approach provides high throughput.	1. Need to improve shift decisions of an algorithm.
3. A Two Way Pattern Matching Algorithm	1. Provides faster pattern matching than existing algorithms. 2. Use of two windows makes it more useful.	1.Time consuming 2. Takes more number of comparisons
4. Naive Pattern matching algorithm	1. Suitable for larger strings 2. Simple algorithm is used	1. Time complexity is not so improved. 2. More memory consumption
5. A2KD string pattern matching algorithm	1. Takes less execution time 2. needs no preprocessing	1. More number of comparisons. 2. Not so efficient compared to other algorithms.

VI. PROPOSED METHODOLOGY

String matching algorithms or string searching algorithms are a dominant class of the string algorithms which aim to find one or all occurrences of the string within a larger group of the text. String matching is further divided into two classes exact and approximate string matching. Exact pattern matching algorithms aims to find one or all occurrences of string within a larger group of text string. In exact pattern matching, pattern is fully compared with selected text window of text string. Pattern matching is one of the major issues in the area of

computational biology. Biologists search database for important information in different directions. An Improved Bidirectional Exact Pattern Matching algorithm is an advanced approach to Bidirectional exact pattern matching. Improved Bidirectional exact pattern matching algorithm compares a given pattern character wise from both sides simultaneously as done in Bidirectional algorithm. The main difference between BD and IBD is that, in case of mismatch or a complete match of the pattern, IBD scans pattern for the rightmost character of the partial text window and scan partial text window for the leftmost character of the pattern.

There are multiple pattern matching algorithms in existence. Each algorithm has its own application. Pattern matching plays an important role in many computer related fields, such as information retrieval, intrusion detection, data compression, content filtering, gene sequence comparison and computer virus signature matching. It is a basic problem in computer science. Pattern matching is one of the major issues in the area of computational biology. Biologists search database for important information in different directions. Pattern matching will continue to grow and need changes from time to time. So, to solve the above problem, there is a greater need for highly efficient algorithm. The proposed methodology fulfills this need.

So, as an efficient algorithm is needed to solve the pattern matching problem, the proposed methodology focuses on the newly developed algorithm which is highly efficient. In the proposed algorithm, an efficient text will be retrieved as an output. The matching pattern will also be a text string. The proposed algorithm which is named as an exact pattern matching methodology using text windows will use input as text strings. The proposed methodology is an improvement in pattern recognisation of an exact pattern matching methodology. It has various advantages as compared to other existing pattern matching techniques.

The proposed algorithm for exact pattern matching methodology using text windows is shown below.

Table 1: Improved Exact Pattern Matching Methodology Using Text Windows

IMPROVED EXACT PATTERN MATCHING ALGORITHM
Step 1: Initialization 1. Initialize dictionaries with appropriate words. **Step 2: Take Input from user** 1. Enter an input as a query **Step 3: Remove stop words from an input query.** **Step 4: Grammar checking** 1. Check grammar for an input query. **Step 5: Extracting ungrammatical words from query** 1. Extract all the ungrammatical word from the query. **Step 6: Apply pattern Matching** 1. Apply pattern matching methodology on extracted ungrammatical words. 2. Match the ungrammatical words with the word dictionaries from multiple users. **Step 6: Retrieval of Efficient word** 1. Fetch an efficient text from any of the dictionary. **Step 7: Finish**

In step 1, initialization is done. In initialization, all words are entered into a dictionary. All the appropriate words are entered into a dictionary. In step 2, input is taken from user as a query. In step 3, all the stop words are removed for simplification. In step 4, grammar checking is done. In this grammar will be checked for each word in a query. In step 5, the ungrammatical words will be extracted. In step 6, pattern matching is applied to each consecutive character in a word. Then match is performed on the various suggestions from the ungrammatical word dictionaries from multiple users.

In step 6, the efficient text is retrieved from dictionary.

This efficient task is also referred as an output.

In this way, the algorithm works. The newly improved exact pattern matching methodology uses text string both for input and output. It provides us more detailed and improved match as compared to existing algorithm as it also refers to dictionaries from multiple users.

This improves its efficiency. As, exact pattern matching methodology has multiple algorithms. But to find an efficient one is really a cumbersome task. So, to select the best algorithm for pattern matching, it must provide some extra features and high efficiency as compared to others. The proposed methodology fulfills the need of an user and provide a simple but efficient approach to use a pattern matching methodology.

VII. POSSIBLE OUTCOMES AND RESULT

The proposed improved exact pattern matching algorithm has better performance than the existing algorithms. It has focused on high efficiency. In the proposed algorithm, an efficient text is retrieved as an output. The matching pattern is also a text string. An exact pattern matching methodology is improved because all the existing algorithms take a single dataset for pattern matching. In our algorithm, multiple datasets are used for the retrieval of efficient word from the query after applying the pattern matching technique. It searches for the possible match among all the datasets . It has various advantages as compared to other existing pattern matching techniques.

VIII. CONCLUSION

This paper presents a newly improved approach for exact pattern matching algorithm. The basic idea of an improved exact pattern matching algorithm is, it scans text window of the text string for all the occurrences of a word in a text query. An Improved algorithm compares a given pattern character wise with selected text window from all the linked dictionaries of an user. Linked dictionaries are created only for the users those who are in the communication link of an admin user. The Improved-Bidirectional EPM algorithm is quite efficient than the existing algorithms, when the pattern length is short as well as long pattern's lengths.

IX. FUTURE SCOPE

There is still need to improve the matching comparisons of exact pattern matching problem. The future scope focuses on to improving the algorithm by enhancing the methodology to support for larger queries too. The dataset used can be improved. Instead of using a small dataset, a global dataset can also be used. The use of

global dataset will drastically increase the performance of a proposed newly improved exact pattern matching methodology using text windows.

REFERENCES

[1] Iftikhar Hussain, Samina Kausar, Liaqat Hussain and Muhammad Asif Khan, "Improved Approach for Exact Pattern Matching (Bidirectional Exact Pattern Matching)", IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 3, No 1, ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784, May 2013.

[2] Radhakrishna.V, B.Phaneendra, V.Sangeeth Kumar, "A Two Way Pattern Matching Algorithm Using Sliding Pattern (Approach to reduce comparisons)", 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), 978-1-4244-6542-2, May 2010

[3] Rami H. Mansi, and Jehad Q. Odeh, "On Improving the Naïve String Matching Algorithm," *Asian Journal of Information Technology*, Vol. 8, No. 1, ISSN 1682-3915, pp. 14-23, May 2009

[4] Iftikhar Hussain, Muhammad Zubair, Jamil Ahmed and Junaid Zaf-far, "Bidirectional Exact Pattern Matching Algorithm," *TCSET'2010*, pp. 29-31 , Feb 2010.

[5] Kartikeya Sinhaz and Aman Duggal, "A Fast Pattern Matching Algorithm Using Two Sliding Windows", Journal of computer science, Vol 6, pp. 245-248, September 2010.