Comparative Study between Various Pattern Matching Algorithms

Pranit Chettri
Sikkim Manipal University
Sikkim Manipal Institute of Technology, Majhitar
Department of Computer Science and Engineering,

Chinmoy Kar
Sikkim Manipal University
Sikkim Manipal Institute of Technology, Majhitar
Department of Computer Science and Engineering

ABSTRACT

Present paper describes the details of the study of the work that has been done in the field of text searching, a sub-division of Natural Language Processing (NLP) till date. The work in this project includes the study and analysis of some of the algorithms devised under this topic, finding the faults or loop-holes and trying to increase the efficiency of these algorithms devised, taking forward the range of work done on it. Experiment is done on the various text search algorithms that have been devised namely Knuth-Morris Pratt Algorithm, Naïve Search Algorithm and Boyer-Moore Algorithm by providing text input of various sizes and analyzing their behavior on these variable inputs. After analyzing and doing the study on these algorithms the results states that Boyer-Moore's Algorithm worked quite well and efficiently than the rest of them when dealing with larger data sets. When working on larger alphabets the Knuth-Morris Pratt Algorithm works quite well. These algorithms do have drawbacks as their efficiency depends upon the alphabet/pattern size. And also this paper describes new pattern matching algorithm that uses delimiter for shifting the pattern while matching.

Keywords

NLP, KMP Algorithm, Naive Search, BM Algorithm.

1. INTRODUCTION

Natural Language Processing is a field of computer science, artificial intelligence (also called machine learning), and linguistics concerned with human (natural) languages[1]. It is the process of extracting the meaningful information from any natural language input and/or producing natural language output. Natural language understanding is considered to be an AI-complete problem because of its requirement of extensive knowledge about the world outside and its ability to manipulate it.

NLP is experiencing rapid growth as its theories and methods are deployed in a variety of new language technologies. The problem of string matching is that there are two strings; one is the text T of length *n* and the other is a pattern string P of length *m* i.e. the string to be matched with the given text string T. It is a very important subject in wider domain of text processing and its algorithms are its basic components used in implementations of practical software under most operating systems. Some of the basic implementations of string matching algorithms are seen in gene sequencing, protein analysis, text editors, digital dictionaries, information retrieval, bibliographic search, question answer applications; Artificial Vision also uses string matching techniques as an integral part of their theoretical and practical tools along with musical technology and computational linguistics. The strings are matched according to shifts. If P occurs with a certain shift s in T then that shift is termed as a *valid shift* otherwise it is known as an *invalid shift*. The single string matching problem is to find the first valid shift with which the pattern P occurs in a given text T. This is shown as follows:

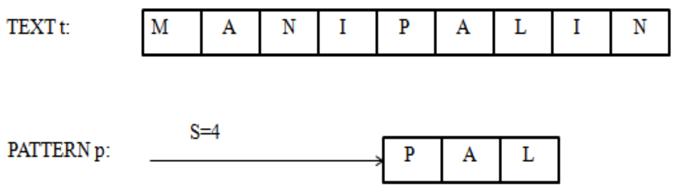

Fig 1: Example of String Matching.

As seen above the matching algorithm uses a concept of window to scan the text. The size of the window depends upon the condition **m≤ n**.

There are four algorithms taken into consideration in this paper and they are Naïve String Matching Algorithm, Knuth-Morris-Pratt Algorithm and Boyer Moore Algorithm. The above mentioned Algorithms were implemented and a comparative analysis was done. The performance measure taken into consideration was the number of iterations required by each algorithm in order to find the shift of P in T.

Experiments are done using Python because it has a shallow learning curve, its syntax and semantics are transparent, and it has good string-handling functionality. As an interpreted language, Python facilitates interactive exploration. As an object-oriented language, Python permits data and methods to be encapsulated and re-used easily. Python comes with an extensive standard library, including tools for graphical programming and numerical processing. The recently added generator syntax makes it easy to create interactive implementations of algorithms.

2. LITERATURE SURVEY

In this section, it tries to mention various knowledge that has been gained from various literature surveys done on the various pattern matching algorithms.

2.1 Naive Search

Also known as proof by exhaustion, proof by cases, perfect induction or brute force method is a method of mathematical proof in which the statement to be proved is split into a finite number of cases and each

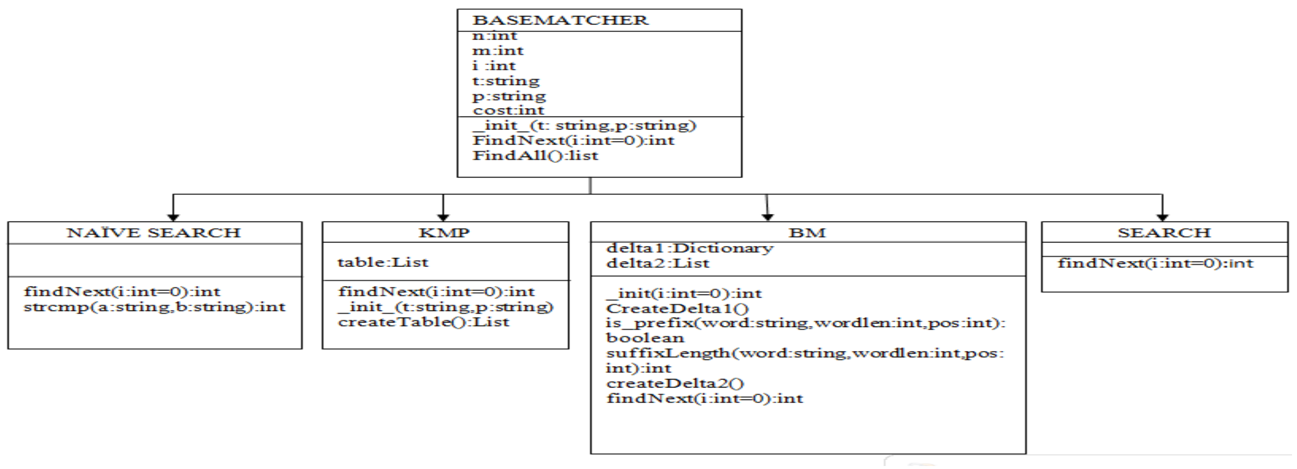

Fig 2: Framework for implementation

case is checked to see if the proposition in question holds. It has no pre-processing phase, needs constant extra space. It always shifts the window by one position to the right. The Naïve Search Algorithm is a brute force matching algorithm with a time complexity of O((n-m+1)m).

2.2 Knuth-Morris-Pratt

Knuth-Morris-Pratt Algorithm bypasses re-examination of previously matched characters and it makes use of prefix table to make a possible shift in the pattern Based on the observation that when a mismatch occurs, the word itself has sufficient information on where the next match should begin. Preprocessing of p gives a partial match table, which indicates where it needs to look for a new match if the current one ends in a mismatch. The preprocessing complexity of this algorithm is given by O(m). The average complexity is given by O(n+m) and the worst case complexity is O((n-m+1)m).

2.3 Boyer Moore

Boyer-Moore preprocesses p, before the commencement of matching procedure. It uses information gathered in preprocessed stage to skip sections of *t*. It is the most efficient contender. It has a worst-case running time of O(n+m) only if the pattern does not occur in the text.

3. IMPLEMENTATION DETAIL

Object-oriented paradigm is followed in this project's 'frame-work' for testing string matching algorithms which is shown in fig.2. Abstract class BaseMatcher is defined from where all other search algorithm classes are derived. This is done because all the string matching algorithms have similar attributes such as the text string t, pattern string p, length of text string n, length of pattern string p, the cost to search, and algorithms like BM and Karp-Rabin require set of alphabets from where text string is derived which is denoted by A.

The algorithm classes inherits BaseMatcher and overrides findNext method, in findNext the algorithm is coded. A function test is defined which accepts an object of BaseMatcher as a parameter and test it. The advantages of this architecture are:

a) Easily extensible; more algorithms can be added by inheriting BaseMatcher.
b) Easily modifiable; minimum change has to be made to reflect a change in the whole project.
c) Easy to maintain; the code tends to become well-structured and not cluttered.

4. PERFORMANCE METRIC

The time required for algorithms to execute was the initial metric, however due to inconsistent results which are shown in figure 2, it was discarded. Every execution of the same algorithm on the same data produced different time for completion. It may be due to the fact that in a multitasking system, due to scheduler assigning time slots to different executing processes on a fast cycle wise basis, there is inconsistency in time taken for completion. Initially the same algorithm was executed on the same data for 10 times and the average time of completion calculated. However eventually it was discarded altogether as unreliable, as average time was also inconsistent.

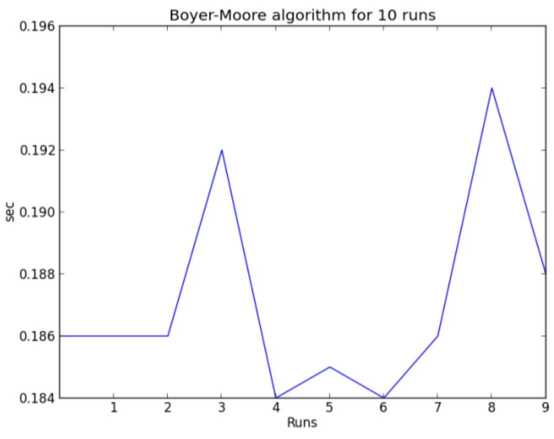

Fig 3: Inconsistent of Boyer Moore Algorithm.

So, to overcome this drawback it makes use number of iteration as performance metric, which has overcome the expectations.

5. DESIGN OF SLSMA

New Pattern/String matching algorithm which is designed makes use of concept of heuristic based on the characteristics of Natural Language Processing, viz. the use of delimiter. This heuristic enables pattern to skip the certain letters in the text string. The newly designed pattern matching algorithm haven't use any pre-processing of text that were done in existing String/Pattern matching algorithm.In this algorithm there are two strings; one is the text T of length *n* and the other is a pattern string P of length *m* i.e. the string to be matched with the given text string T.

i= position , where i<n-(m-1)

n= Length of text string

m=Length of pattern string

p=Pattern string

t=Text string

delim =set of delimiters

5.1 Flowchart

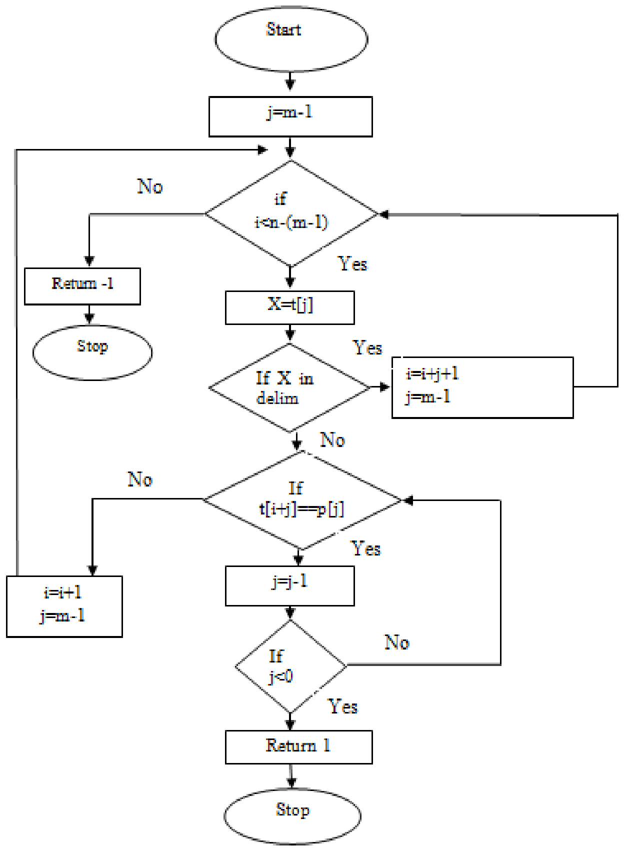

Fig 4: Flowchart for SLSMA

5.2 Method

Let us take text string "a big book" and pattern string as book. Then according to flowchart:

1. Iteration 1:

i=0
j=3
T: a big book
P: book
T[3] is "i" and P[3] is "k"; "i" ≠ "k" , "i" does not exist in 'delim', therefore i🡐i+1.

2. Iteration 2:
i=1
j=3
T: a big book
P: book
T[4] is "g" and P[3] is "k"; "g" ≠ "k" , "g" does not exist in 'delim', therefore i🡐i+1.

3. Iteration 3:
i=2
j=3
T: a big book
P: book
T[5] is " " and P[3] is "k"; " " ≠ "k" , " " exists in 'delim', therefore i🡐i+j+1.

4. Iteration 4:
i=6
j=3
T: a big book
P: book
T[9] is "k " and P[3] is "k"; "k" = "k" , therefore j🡐j-1.

5. Iteration 5:
i=6
j=2
T: a big book
P: book
T[8] is "o" and P[2] is "o"; "o" = "o" , therefore j🡐j-1.

6. Iteration 6:
i=6
j=1
T: a big book
P: book
T[7] is "o" and P[1] is "o"; "o" = "o" , therefore j🡐j-1.

7. Iteration 7:
i=6
j=0
T: a big book
P: book
T[6] is "b"and P[0] is "b"; "b" = "b",therefore j🡐j-1.

Now, j = -1
Therefore, P is found in 'i' position of T, which is 6 and total of 7 iterations were required to find the solution.

6. EXPERIMENTS AND RESULTS

This section provides the analysis and the results got after doing a comparative study on the various text search algorithms like KMP Algorithm, Naïve Search Algorithm and BM Algorithm giving the details of their individual performances based on various existing real time books taken as inputs for datasets which vary in size so that a proper study of the results could be produced by the various algorithms.

This paper have discussed the complexities, provided below are the practically experimented graphs of performances shown by the algorithms when implemented on the real time books that exist. It has taken into consideration two famous books of the world namely Gulliver's Travels by Jonathan Swift which acts as a text string of size 590 KB and the other is Iliad of Homer by Homer which is of size 1.11 MB. It has taken four pattern strings into consideration in such a way that the pattern string:-

- Exists in the beginning of text
- Exists in the middle of the text
- At the end of the text

- Does not exist in the whole text.

The performance graphs are given below:-

It takes into account the book “Gulliver’s Travels” by Jonathan Swift which acts as a sample text of 590 KB and got the following graphs for the above four different case studies:-

- When the pattern string to be searched for was provided as “Jonathan” which exists in the beginning of the book it produces the following result in the performance analysis graph:

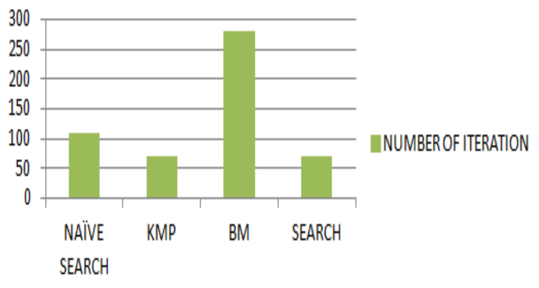

Fig 5: Graph showing performance of various algorithm when pattern is at the starting of the text.

When the pattern string to be searched for was provided as “Glubbdubdrib” which occurs in the middle of the story it produces the following result in the performance analysis graph:

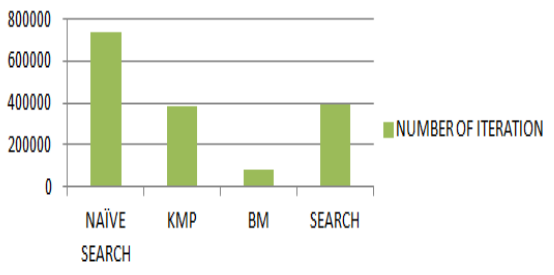

Fig 6: Graph showing performance of various algorithms when pattern is at the middle of the text.

When the pattern string to be searched for was provided as “Reflectors” which occurs at the end of the book it produces the following result in the performance analysis graph:

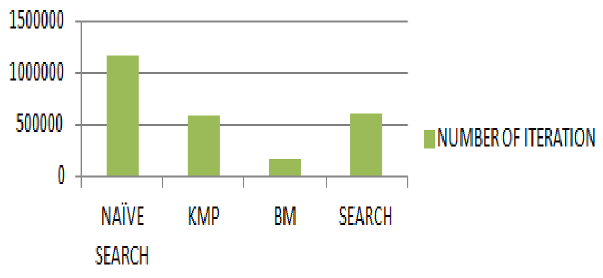

Fig 7: Graph showing performance of various algorithms when pattern is at the end of the text.

The pattern string to be searched for was provided as “Sikkim” which does not occur at all in the book. This was done in order to make the algorithms work in their worst case.

It produces the following result in the performance analysis graph:

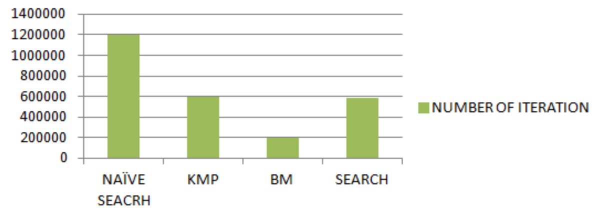

Fig 8: Graph showing performance of various algorithms when pattern is not present of the text.

Table 1: Results after Comparisons

ALGORITHM / POSITIONS	NAÏVE SEARCH (in iterations)	KMP (in iterations)	BM (in iterations)	SLSMA (in iterations)
AT START	120	70	275	60
AT MIDDLE	7,30,000	3,90,000	7,00,000	4,00,000
AT LAST	11,70,000	5,90,000	1,70,000	6,00,000
NOT PRESENT	12,00,000	6,00,000	2,00,000	6,10,000
TOTAL (in iterations)	31,00,120	15,80,070	10,70,275	16,10,060

7. APPLICATIONS

There are various fields where the application of string matching algorithm is deployed. Following are some of the fields of Pattern/String matching algorithm.

7.1 Information Retrieval

Despite the use of indices for searching large amounts of text, string searching may help in an information retrieval system. For example, it may be used for filtering of potential matches or for searching retrieval terms that will be highlighted in the output.[6]

7.2 Retrieving Musical Patterns

Given a search space composed of sequential stream of n elements in which the elements are from a set A. String matching algorithms can be used to find the occurrence of certain musical patterns from the database. The musical notes are retrieved by QBE (Query by Example) approach. The best scheme for this problem is Levenshtein distance with Jaccard similarity. As the Jaccard similarity performs excellent in passing a query when a pitch change scenario is selected [6].

7.3 Molecular Biology

Gene sequence is a string derived from the set of alphabets {a,c,g,t} where a stands for adenine, c for cytosine, g for guanine and t for thymine. A string matching algorithm can be used to find a particular subsequence in a gene sequence[3].

7.4 Natural Language Processing:

String matching is extensively used in NLP to search for occurrence of words or to search for supporting words which describes the context in which the particular word is being used.[6]

7.5 LZgrep Tool

Boyer Moore technique is used for string matching over LZ78 and LZW compressed texts. This is done directly on the compressed text hence speeds up the best decompress-then-search approach by upto 50% [6].

7.6 Medical Texts

The Boyer Moore Horspool algorithm achieves the best overall results when used with medical tests. This algorithm performs at least twice as fast as the other algorithms tested [6].

7.7 Network Intrusion Detection System:

The ability to search through the packets and identify content that matches known attacks is very important for which string matching algorithms are used. This requires exact pattern matching technique for which Boyer-Moore is used [6].

7.8 Polymorphic String Matching

This technique refers to fusions of some of the string matching algorithms which when done will increase the efficiency as some of the time consuming parameters need not be used. Tree data structure is used in data representation. One example is KMP and Boyer-Moore fusion. Their combination completes the task in amortized constant time and cost is also equal to equality check cost. The quadratic time is dropped and features of both the algorithms are combined and used for producing a better functional algorithm.[6]

8. CONCLUSION AND FUTURE SCOPE

In this paper a detailed description is provided about the different string matching algorithms that has been studied and analyzed with different input texts provided in the form of books and the performance of these algorithms have been show in the form of graphs in terms of number of iterations each algorithm uses to find the pattern string provided. Therefore, it’s concluded that Boyer-Moore’s Algorithm works most efficiently than the other algorithms under study but since BM Algorithm has to maintain a dictionary containing the alphabets in the set from where the test string is derived, the space complexity tends to be greater than other algorithms. The New Pattern matching algorithm or SLSMA that have design tends to perform well when the text string is short and there is use of delimiters. And it produces better result than existing string matching algorithm if pattern occurs at the starting of the text. The New pattern matching algorithm or SLSMA can perform better if it makes use of concepts of cellular automata and longest common subsequence.

9. REFERENCES

[1] Natural language processing,online: http://en.wikipedia.org/wiki/Natural_language_processing, Access Date: 23th May,2015.

[2] Koloud Al-Khamaiseh, Shadi ALShagarin"A Survey of String Matching ", Int. Journal of Engineering Research and Applications, ISSN : 2248-9622, Vol. 4, Issue 7(Version 2), pp.144-156,July 2014.

[3] Pandiselvam.P,Marimuthu.T ,Lawrance. R,"A Comparative Study On String Matching Algorithms Of Biological Sequences”Deptt of Computer Applications,Ayya Nadar Janaki Ammal College, India,jan 2014.

[4] Hussain I., Kausar S., Hussain L., and Asifkhan M.”,Improved Approach for Exact Pattern Matching, International”, Journal of Computer Science Issues, Vol.10, Issue 3, No.1,2013.

[5] Jain P., Pandey S., “Comparative Study on Text Pattern Matching for Heterogeneous System”, International Journal of Computer Science and Engineering Technology, ISSN: 2229-3345, Vol.3 No.11 Nov 2012.

[6] Singla N., Garg D.,“String Matching Algorithms and their Applicability in various Applications”, International Journal ofSoft Computing and Engineering, ISSN: 2231-2307, VolumeI,Issue-6, January 2012.

[7] R.S. Boyer and J.S. Moore, “A Fast String Searching Algorithm”, SRI International, 1977.

[8] Donald Knuth, James H. Morris and Jr. Vaughan Pratt, “Fast pattern matching in strings”, SIAM Journal on Computing, 1977.

[9] Richard M. Karp, Michael O. Rabin, “Efficient randomized pattern-matching algorithms”, IBM Journal of Research and Development, 1987.