ISSN: 2321-7782 (Online)
Volume 3, Issue 9, September 2015
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Data De-duplication using large scale pattern

Deotarse Mayuri[1]
Copmuter department
JSPM(ICOER) Wagholi
Pune, India

Murte Vijaya[2]
Copmuter department
JSPM(ICOER) Wagholi
Pune, India

Chaudhari Revati[3]
Copmuter department
JSPM(ICOER) Wagholi
Pune, India

Yoge Kishor[4]
Copmuter department
JSPM(ICOER) Wagholi
Pune, India

Abstract: The general purpose of string matching algorithm are finding pattern in given large data. Basically string matching algorithm used in text mining, books, paragraph, sentences, data de-duplication, data security, computer science, computer networking, detecting plagiarism etc. In this paper we shortly represent on string matching algorithm .string matching algorithm describe in following part: Basic string matching algorithms, BWT. And suffix array is better than suffix tree in many application.

Keywords: Suffix tree, Pattern matching, suffix array disk based algorithms.

I. INTRODUCTION

String matching algorithm are used to finding the pattern in given large data. Also known as exact string matching, string searching, text searching. In this paper string matching algorithm two main part : exact string matching, Approximate string matching. Also used many algorithm, those algorithm used only string matching.

Suffix tree introduce by Weiner and time and space complexity O(n), in memory construction algorithm. Suffix tree is important in data structure for processing of string and this tree support efficient solution to solve many problem involving in pattern matching and large string pattern discovery, such as found in computational biology. some problem suffix tree excel in pattern matching: given an M character pattern p (0 1……..M-N)and N character text t(0 ….N-1),if the p occurrences the reports position in t .when suffix tree is build from t then pattern could be locate in the text in time O (M + occ) occ: No occurrences of pattern in given text. Suffix tree is display discovery of generic pattern, for example they use to locate repeated substring in text in optimal time.

Suffix tree layout on disk or suffix array in memory is define primarily by construction algorithm use them and do not used in application and any algorithm. In suffix tree more efficient way to locate a pattern is travels in top down.

The fast growth in size number Popularity of database and also other type of information, has been display then suffix redoubled. In suffix tree any element find out that time traversals from root node. This problem of suffix tree is typically required 12Nbyte memory to represent text of N character, that character assume in t and p (t text, p pattern) ,are draw from a alphabet that is ASCII large alphabet require more space. Suffix tree access is solver The research in another development has evolution of suffix array in data structure . It has recent show that if the argument relatively small auxiliary array then it has recently show , suffix tree is equivalent to suffix array. Suffix array introduce by Mender and Myers. The algorithm map one data structure can map to other this is called as enhance-suffix array, enhance-suffix array is array base completely, enhance-suffix array used for improve the locality of references. Suffix array is an space efficient than suffix tree.

Suffix array use several algorithms for memory exist. it is use for speed improvement over suffix tree in memory. Suffix tree work poorly situated implemented on disk because of random access memory. Suffix array featly work foe random access.

II. String Matching Algorithm

String matching algorithm is used for the compilation program in text processing. It is an vital activity. In text processing string matching is important activity. String matching is nothing but to finding one or more occurrences of string in given data this process is called as pattern. String matching algorithms also called as pattern matching algorithms.

***A. Exact String Matching*:** Exact String Matching means do not find the matches of occurrences of a pattern in a given text .It is a comparison between different characters and it is based upon the different methods like hashing string matching, bit-parallelism and deterministic finite automata.

***1. Hashing string matching*:** Hashing string matching is used for the shifting substring search.

» ***Rabin–Karp algorithm*:** By using hash function testing of the equality pattern to substring in the text and hash function used for they convert string into the numeric value.

» ***Wu-Mender (WM) Algorithm:*** Wu-Mender (WM) Algorithm based on Boyer-Moore algorithm. Preprocessing uses three tables hash, prefix and shift. This algorithm shift the bad character this character consider from the given text instead of given block size B and character one by one consider. They use minimum length of pattern then this algorithm gives good performance. The occurrences of text or patterns to assign the hash value and the shift table check the corresponding hash value. Check the value and if value is greater than zero then sift the text.

2. Classical method:

There are types of string matching algorithms

1. Naive pattern matching algorithm
2. Boyer Moore algorithm
3. KMP algorithm

» ***Naive pattern matching algorithm***: Naive pattern matching algorithm used brute force approach. It is a straightforward approach .This approach is used for solving the problem related to string. This algorithm is pattern to check left to right and one character is mismatch then shifts the pattern one place at right side. If one character match pattern with string then next character match pattern with string.

» ***Boyer Moore algorithm***: Boyer Moore algorithm scans the character pattern in the Right to Left. This is nothing but “Looking glass heuristic” and “Bad character algorithms”. If match is not found then the shift pattern by using following two condition :

1. IF L+1<j then shift j-1
2. If L+1>j then shift 1.

Where, L= last character of text, J=index of pattern

The worst case running time is O (m n+| Σ|).

Advantages are: this algorithm is faster than Naive algorithm. It is an efficient and it is used for the long text of data. Disadvantages are: this algorithm implementation and understanding complex.

KMP matching algorithm: Pattern matching algorithm like naïve algorithm and Boyer Moore algorithms in that algorithm compare the pattern character in the text that time pattern character does not match in the text means occurrence the mismatch these all pattern throw the information and again create new set of pattern and those pattern match in the text. Pattern are matches each character of next position in text. Knuth-Morris-Pratt avoids the repeatasion of characters. In this algorithm the basic concept build prefix array and this called as π array. By using the prefix and suffix information of pattern to build the π array. The overlapping prefix and suffix occurred in KMP algorithm. Worst case of KMP is O (m + n). Advantages are: The avoid waste of information during the scan of text. Running time of algorithm is very fast and good for large file. Disadvantages are: If size of alphabets increases it do not work properly and more chances of occurrences.

The meaning of proper prefixes and proper suffixes are:

1. ***Proper prefix***: The string of character cut one or more occurrences, cut off the end. For example “RANISONI” string prefixes are as follow “R”, “RA”, “RAN”, “RANI”, ”RANIS”,”RANISO”,”RANISON”, ”RANISONI”.
2. ***Proper suffix***: The string of character cut one or more occurrences, cut off the beginning. For example “RANISONI” string suffixes are as follows “I”, “NI”, “ONI”, “SONI”,”ISONI”,”NISONI”,”ANISONI”,”RANISONI”.

B. Approximate string matching: Approximate string matching is use fuzzy string searching technique, This technique use for find strings and these strings are matches with approximately pattern. Approximate string matching algorithms are divided into two categories: off-line and online. In online technique text cannot be process but pattern processed before searching. It is improved technique. In this technique implementing a several function such as: Hamming distance function, Levenshtein distance function. Generally in string matching application perform following three operations:

a) One substation character with another single character.

b) In given string one character deletes.

c) In given string one character inserts.

Hamming distance function: If two strings of equal length then finding the positions of mismatching characters. Approximately string matching algorithm is also called as hamming distance matching algorithm and also called as K-mismatches.

Levenshtein distance function: If transferring one string to another string then performs operation are inserting, deleting and substitution. It is also known as d Levenshtein distance string matching and K-differences. The uses of approximate string matching are: low quality of text, heterogeneous data, spelling mistake in pattern or text. Basically approximation string matching based on different method deterministic finite automata, bit-parallelism, classical/dynamic programming, counting and filtering string matching algorithms.

1. Classical/dynamic programming Method:In dynamic programming given problem is divided into sub problem and this sub problem is again subdivided and find optimum solution. It is used for find shortest path distance and Depth first algorithm. Classical method used for character comparison.

» ***Burst force algorithm***: . It is a straightforward approach .This approach is used for solving the problem related to string. This algorithm is pattern to check left to right and one character is mismatch then shifts the pattern one place at right side. If one character match pattern with string then next character match pattern with string.

» ***Diagonal transition algorithm:*** It is basically used for developing new algorithm. It is used in dynamic programming array monotonic increase.

» ***Sellers Algorithm***:It is used in dynamic programming method and it is find out occurrences pattern in given text.

2. ***Deterministic Finite Automata Method***: Deterministic Finite Automata Method is also called as Non deterministic Finite Automata Method . It is simple method for any programming language suppose in programming language alphabet A compare to remaining 25 alphabets like A=B, A=C….. up to A=Z .but in Deterministic Finite Automata

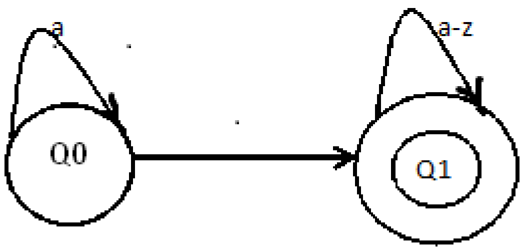


fig.1 Deterministic Finite Automata

- ***Kekkonen's algorithm:*** Kekkonen's algorithm is used for computational biology problems and numerous string processing. Suffix tree used in many application but not easily implement the code. The purpose of suffix tree bridge between complete working code implementation and theory. In this algorithm discuss about step by step theory implementation and discuss about code implementation by using the brute force algorithm.

- ***Bitmap Algorithm*** (Wu–Mender–Myers Algorithm)The Bitmap algorithm is also known as shift or, shift and, or Baez-Yates-Gannet algorithm. It is an under the approximate string matching algorithm. If the text are approximate match to pattern then term is defined under L even stein distance. This algorithm is used for exact searching.

C. ***Bit-Parallelism method***: In this approach perform some parallel operation when a nondeterministic finite automaton is converted into deterministic finite automata. It consist basically three algorithms: Shift-Or algorithm, Kekkonen's algorithm for k-differences matching, Myers' bit vector algorithm.

- ***Shift-Or (SO) Algorithm:*** This algorithm is used to maintain all prefixes of P and this p matches with suffix of text read .Each new text character use in bit parallelism to update. Indices run in left to right pattern. But bit numbers are MSB to LSB that is right to left.

- ***Kekkonen's algorithm for k-differences matching:*** It is a properties of dynamic programming matrix. If normally pattern is not match with text then each column read from top to bottom. By using the properties of lacto they avoid working of subsequent cells so this algorithm maintain index lacto pointing to last active cell and updates.

- ***Myers' bit vector algorithm:*** It is a process of dynamic programming using bit-parallelism and this is used for approximately string matching algorithm. Parallelism is the difference between consecutive column and rows of dynamic programming matrix. By using text and pattern dynamic programming assign the index.

III. Indexing using a Condensed BWT

Structure of Burrows-Wheeler algorithm:

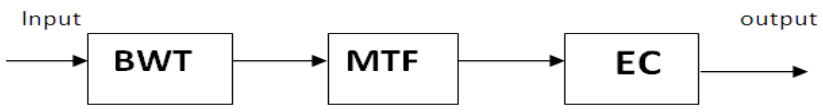


fig2. Structure of Burrows-Wheeler algorithm:

Burrows-Wheeler algorithm consists of three stages:

- The first stage of burrows wheeler algorithm is BWT. BWT is called as Burrows Wheeler Transform.BWT is used for sorting characters of the input identical character close to each other.

- The second stage of burrows wheeler algorithm is MTF. MTF stand for Move to front transform. It is used to assign the global index value which characters input find in local context.

» Third stage is EC. EC is nothing but Entropy Coding. In above two stages those data are used this data are compress.

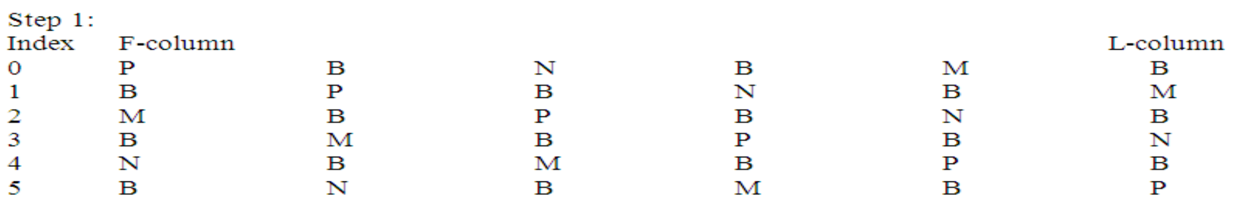
A. BWT: The aim of BWT is sort the input character and these input character identically close to together. Process steps:

1. Consider order the n input times among themselves, each row rotate by one character in right side and compared to the previous row. n= Length of input
2. Sorting the rows using lexicographically

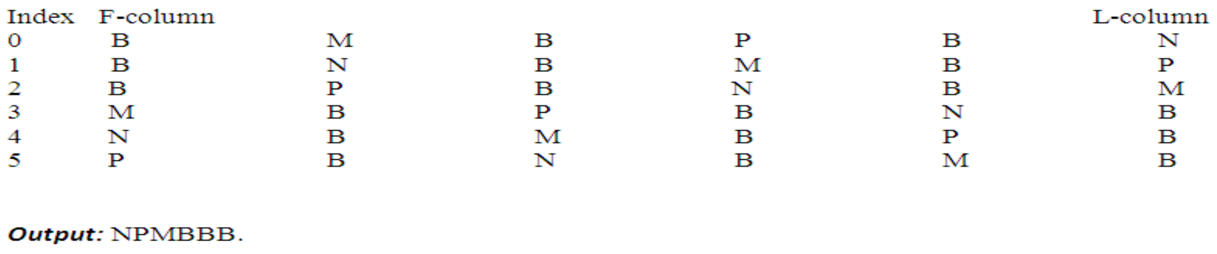
The output of this stage is the L-column

The index value of the sorted matrix that contains the original input,

We explain the procedure step by step with PBNBMB as example input.


Step 1:

Index	F-column					L-column
0	P	B	N	B	M	B
1	B	P	B	N	B	M
2	M	B	P	B	N	B
3	B	M	B	P	B	N
4	N	B	M	B	P	B
5	B	N	B	M	B	P



Index	F-column					L-column
0	B	M	B	P	B	N
1	B	N	B	M	B	P
2	B	P	B	N	B	M
3	M	B	P	B	N	B
4	N	B	M	B	P	B
5	P	B	N	B	M	B

Output: NPMBBB.

In above example that in the output the same characters are close together as in the input. This output consider as input to the next stage move to front.

IV. Conclusion

In this paper, the various string matching algorithms were studied such as KMP algorithm is to implement due to it need not to move sequence of input in backward, Rabin Karp algorithm, purpose of this algorithm detect the plagiarism, Brute Force algorithm does not require the text or the pattern preprocessing, again many algorithm used in this paper. In future the purpose of string matching algorithm for finding data de-duplication, speedy and efficient.

References

1. U. Member and G. W. Myers, “Suffix arrays: A new method for on-line string searches,” SIAM J. Comput., vol. 22, no. 5, pp. 935–948, 1993.
2. R. Sinha, S. J. Puglisi, A. Moffat, and A. Turpin, “Improving suffix array locality for fast pattern matching on disk,” in Proc. ACM SIGMOD Int. Conf. Manage. Data, Vancouver, BC, Canada, 2008, pp. 661–672.
3. A. Moffat, S. J. Puglisi, and R. Sinha, “Reducing space requirements for disk resident suffix arrays,” in Proc. 14th Int. Conf. DASFAA, Brisbane, QLD, Australia, 2009, pp. 730–744.
4. V. Mäkinen and G. Navarro, “Compressed compact suffix arrays,” in Proc. Symp. CPM, Istanbul, Turkey, 2004, pp. 420–433.
5. L. Colussi and A. De Col, “A time and space efficient data structure for string searching on large texts,” Inform. Process. Lett., vol. 58, no. 5, pp. 217–222, 1996
6. B. Phoophakdee and M. J. Zaki, “Genome-scale disk-based suffix tree indexing,” in Proc. ACM SIGMOD Int. Conf. Manage. Data, Beijing, China, 2007, pp. 833–844.
7. P. Smith, “Experiments with a very fast substring search algorithm”, Softw. Pract. Exp., Volume 21, No. 10, pp. 1065-1074, 1991.
8. L. Colussi, “Correctness and e efficiency of the pattern matching algorithms”, Information and Computation, Volume 95 Issue 2, Dec. 1991.
9. T. Raita, “Tunning the Boyer-Moore-Horspool string searching algorithm”, Software- Practice and Experience, Volume 22, No. 10, pp. 879-884, 1992.
10. M. Crochemore, A. Czumaj, L. Gasieniec, S. Jarominek, T. Lecroq, W. Plandowski, and W. Rytter, “Speeding Up Two String Matching Algorithms”, Algorithmica, Volume 12, No. 4-5, pp. 247-267, 1994