Study on Applications of String Searching and Matching Algorithms

P.Sundari1, S.Deepasamili2
Assistant Professor1
PG & Research Department of Information Technology
Government Arts College (Autonomous), Coimbatore-18.
M.Phil. Research Scholar2
PG & Research Department of Computer Science
Government Arts College (Autonomous), Coimbatore-18.

Abstract: Many organizations gather huge amount of data to maintain their business and decision making process. The data gathered from different sources can have data features problems. These types of problem become familiar when different databases are combined. The data in the combined structure need to be cleaned for appropriate decision making. Cleansing of data is one of the major critical problems. In this survey, focus is on one of the major issues that varies duplicate detection and search process using string searching and string matching algorithms.

Keywords: Duplicate detection, string searching, string matching.

I INTRODUCTION

Discovering database records that are partially duplicates, but not accurate duplicates, is an significant task. Dataset may have duplicate records with the same real-world entity because of data entry mistakes, unstandardized contractions, or variation in the complete schemas of records from numerous dataset. In real time appliance, classification of records that denote the similar entity is a main contest to be solved. Such records are represented as duplicate records. This survey offered analysis of the literature on duplicate record detection using string searching and matching algorithm.

II BASIC CONCEPT:

STRING MATCHING:

String matching is a technique to locate occurrence of a pattern string within one more text string. Given a text T [0...n-1] and a pattern P [0...m-1] where $m \leq n$, find all occurrence of the pattern within the text.

Example: T = 010001100010110001 and P = 0001, occurrence they are:
First occurrence starts at T [2], Second occurrence starts at T [7], and Third occurrence starts at T [14].
To discover pattern within a text various string matching algorithms are used.

III RELATED WORKS

A.RABIN KARP ALGORITHM:

It is a string search algorithm which evaluates string's hash values, rather than the string itself. For effective, the hash value of the next position in the text is simply calculated from the hash value of the present position.

Rabin karp string searching algorithm utilizes hashing to conclude any one set of pattern strings in a text. The Rabin-Karp string searching calculates a hash value for the pattern, and for each M-character correlate text to be calculated. If the hash values are unequal, the algorithm will compute the hash value of next M-character sequence. Then the hash values are equal, the algorithm will calculate the pattern and the M-character sequence. In this method, there is only one comparison for each text subsequence, and character matching is only required when hash values match. Rather than following additional complicated skipping, the Rabin–Karp algorithm looks testing of equality and the pattern to the substrings in the text by using hash function [8].

Hash value computation:

The key to the Rabin–Karp algorithm's performance is the effective computation of hash value of the successive substrings of the text. The Rabin finger print is a familiar and efficient rolling hash function. The Rabin fingerprint treats eash substring as a number in some base, the base being generally a large prime. For example, if the substring is "hi" and the base is 101, and then the hash value would be $104 \times 101^1 + 105 \times 101^0 = 10609$ (ASCII of 'h' is 104 and of 'i' is 105). Systematically, this algorithm is only same to the true number in a non-decimal system description, because for the example we could have the "base" less than one of the "digits". Let hash function for a much more detailed discussion. The important benefit achieved by utilizing a rolling hash such as the Rabin fingerprint is that it is possible to calculate the hash value of the next substring from

© 2016 IJAIR. All Rights Reserved

the earlier one by doing only a constant number of operations, independent of the substring's lengths.

For example, if we have text "abracadabra" and we are searching for a pattern of length 3, the hash of the first substring, "abr", utilizing 101 as base is: ASCII a = 97, b = 98, r = 114. Hash ("abr") = $(97 \times 101^2) + (98 \times 101^1) + (114 \times 101^0) = 999{,}509$.

We can then calculate the hash of the next substring, "bra", from the hash of "abr" by subtracting the number further added for the first 'a' of "abr", example 97×101^2, multiplying by the base and adding for the last a of "bra", i.e. 97×101^0. Like so: base old hash old 'a' new 'a' Hash ("bra") = $[101 \times (999{,}509 - (97 \times 101^2))] + (97 \times 101^0) = 1{,}011{,}309$. If the substrings in question are lengthly, this algorithm accomplish great savings compared with many other hashing schemes.[8]

B. BOYER-MOORE ALGORITHM:

It was built-up by Robert S. Boyer and J. Strother Moore in 1977. It is an efficient string searching algorithm that is the standard benchmark for practical string search study. The BM algorithm is consider the most effective string-matching algorithm in commen applications, for example, in text editors and commands substitutions. It woks the highest when the alphabet is moderately sized and the pattern is comparatively long. During with testing of a possible placement of pattern P against text T, a dissimilar of text character T[i] = c with the corresponding pattern character P[j] is handled as follows: If c is not represent anywhere in P, then shift the pattern P completely past[i]. Otherwise, shift P until an occurrence of character c in P gets joined with T[i]. As per the study the Boyer Moore algorithm is the best for the string.[6]

For example,
Let Input: MRRQRKKTETDDMAREWQLMS
Pattern: KKT
After the execution of Boyer Moore algorithm
Input: MRRQR***KKT***ETDDMAREWQLMS
| | |
Pattern: ***KKT***
Position ^

C. KNUTH- MORRIS-PRATT ALGORITHM:

The Knuth-Morris-Pratt Algorithm (KMP) was formed by D. Knuth, J. Morris and V. Pratt in 1974. Knuth, Morris and Pratt built-up a linear time algorithm for the string matching problem. In this algorithm, the pattern is evaluated with the text from left to right. In case of a variation or whole match it utilizes the notion border of the string. It decreases the time of searching compared to the Brute Force algorithm. It assures that a string search will not need more than N character comparison. Knuth-Morris-Pratt algorithm's asymptotic time complication is O(n).The sprint time of KMP algorithm is comparative to the time wanted to read the characters in text and pattern. In additional words, the most horrible–case flow time is O (m+n) and it requires O (m) extra space.[3]

D. BRUTE FORCE ALGORITHM:

It is also recognized as evidence by tiredness, also known as evidence by cases. The brute force method is a method of mathematical proof in which the report to be verified is divided into a finite number of cases and every case is tested to see if the proposal in query holds. Evidence by exhaustion has two stages: evidence that the cases are exhaustive; i.e., that every case of the statement to be established matches situations of (at least) one of the cases and an evidence of each of the cases. [4]

E. ALGORITHM USING LEVENSHTEIN DISTANCE:

It is a metric for quantify the quantity of variation between two sequences (i.e. an edit distance). The term edit distance is often utilized to refer particularly to Levenshtein distance. The Levenshtein distance between two strings is explained as the minimum number of edits wanted to convert one string into the other, with the acceptable edit process being insertion, deletion, or replacement of a particular character.[5]

IV APPLICATONS:

String Prefix Matching Problem:

This represents the matching of the prefixes of the pattern and the text. It also tests the maximum prefix of some specified sequence text. This arises at the starting of the patterns. It also contains preprocessing of the pattern. KMP algorithm and deterministic sequential comparison model are applied to solve this problem. This can be done by conveying the lower and the upper bounds of the prefix to be matched.

Retrieving Music Pattern from Musical Database:

When musical note from musical database are to be recovered then we want string matching. The four parallel techniques used for this are edit distance, dice similarity, jacquard similarity and cosine similarity. The musical notes are retrieved by Query by Example approach. So the greatest scheme for this technique is Levenshtein distance with jacquard

© 2016 IJAIR. All Rights Reserved

similarity. This is an inexact music search technique. As the jacquard similarity execute exceptional in passing a query when a pitch change scenario is selected [11].

Network Intrusion Detection System:

This problem includes accurate pattern matching problem. This is open source Intrusion detection system snort. It decrease computational time and higher order of context matching is executed. Boyer Moore algorithm is utilized to resolve this as it wants precise matching of the certain pattern. To accomplish this tree data structure is utilized in adapted algorithms that translate the bad character shift to good prefix shift and resulting in far better performance.

String matching in detecting plagiarism:

Organization of huge collection of simulated data in fundamental environments is significant for several systems that give data mining, reflecting, storage, and content distribution. In its easiest form, the documents are formulated, duplicated and modernized by emails and web pages. Although redundancy may enhance the reliability at a level, unreserved redundancy aggravates the recovery functions and might be ineffective if the revisited documents are obsolete.

This suggests new plagiarism detection techniques by using Karp-Robin algorithm and String Matching algorithm. Here data reliance expression file, take out keyword and utilize twin algorithm technique which conquer all problems of matrix, parallel hash value as well as string matching, which detects plagiarized programs or documents by using hash function.

Experiments have well verified its effectiveness over existing tools and it is appropriate in practice.[10]

Clone Detection System:

Code duplication is a usual problem found in software development. It could produce various clones. Clone is a block of code. It reproduces many time on the source code. The existence of clone is highly probable to intensify the risk on software progress. Technique for detecting clone includes textual, lexical, syntactic, and semantic approach. In this algorithm, we estimate by utilize various aspects based on the condition of every pair. We progress a novel method to detect clone by using Rabin-Karp parallel algorithm. The algorithm is more efficient than the Rabin-Karp algorithm. In cases of estimation, we construct a detecting tool competent of processing source code in both lexical and syntactic manner. We estimate the performance of the proposed method. To do so, we contrast parallel Rabin-Karp to Traditional Rabin-Karp. The result express parallel Rabin-Karp could gain best performance.[9]

ALGORITHM & ITS CHARACTERISITCS:

ALGORITHM	COMPARISON ORDER	CHARACTERISTICS
Rabin karp	Left to right	Use hashing function, very effective for multiple patterns matching, ID matching.
KMP	Left to right	Independent of alphabet size, use the notion of border of the string, increases performance, decrease delay and decrease time of comparing.
Brute force	Not relevant	Use one by one character shift. Not an optimal one.
Boyer Moore	Not relevant	Use both good suffix shift and bad character shift.

V CONCLUSION:

The algorithms for string searching may not be best optimal algorithm but better than the usual algorithms. Rather than utilizing each algorithm to every application one application is represented with certain optimal algorithm. Then it has been noticed that main applications uses Boyer Moore, BMH or KMP algorithms for their efficiency and effectiveness and other applications utilize the basics of these algorithms for their functionalities as the KMP algorithm has less time complexity and Boyer Moore algorithms has preprocessing time with less complexity. The Robin Karp algorithm will produce result of precision value up to 85% and above as well as recall value. It is also able to minimize failed detection percentage around 10%. Other algorithms depend upon the type of input and are effective for particular application.

REFERENCES:

1. Boyer Moore Word Search Algorithm Definition Retrieved from Wikipedia

2. Dany Breslauer, Livio Colussi and Laura Toniolo, 'Tight Comparison Bounds for the String Prefix Matching Problem', Stiching Mathematic Centrum, Amsterdam, 1-9, 1992.

© 2016 IJAIR. All Rights Reserved

3. Kunth Moris Word Search Algorithm Definition Retrieved from Wikipedia
4. Minal Suthar Amit Patel Shivali Shah" A Survey Paper on String Matching IJSRD - International Journal for Scientific Research & Development Vol. 3, Issue 05, 2015.
5. Nimisha Singla, Deepak Garg "String Matching Algorithms and their Applicability in various Applications" International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-I, Issue-6, January 2012 218.
6.Pandiselvam.P, Marimuthu.T, Lawrance. R, A comparative study on string matching algorithms of biological sequences.
7. Ramazan S. Aygün "structural-to-syntactic matching similar documents", Journal Knowledge and Information Systems archive, Volume 16 Issue 3, August 2008.
8. Rabin Karp Word Search Algorithm Definition Retrieved from Wikipedia.
9. Rahadian Dustrial Dewandono , Fahmi Akbar Saputra , Siti Rochimah"Clone Detection Using 8. 8. Rabin-Karp Parallel Algorithm" The Proceedings of The 7th ICTS, Bali, May 15th-16th, 2013 (ISSN: 9772338185001)
10. Sonawane Kiran Shivaji, Prabhudeva S "Plagiarism Detection by using Karp-Rabin and String Matching Algorithm Together" International Journal of Computer Applications (0975 – 8887) Volume 116 – No. 23, April 2015.
11. Vidya SaiKrishna, Prof. Akhtar Rasool and Dr. Nilay Khare" String Matching and its Applications in Diversified Fields" IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 1, January 2012.
12. Wikipedia The free Encyclopedia en.wikipedia.org/wiki/String_searching_algorithm,
13. Yu-lung Lo Chien-Chi Huang, 'Fault Tolerant Music Retrieval by similar String Matching', National Science Council of ROC Grant NSC98-2221-E-324-027,1.

© 2016 IJAIR. All Rights Reserved