COMPARISON OF STRING MATCHING ALGORITHMS IN WEB DOCUMENTS

H. Nusret Buluş
Namık Kemal University

Erdinç Uzun
Namık Kemal University

Alpay Doruk
Namık Kemal University

Abstract
String matching algorithms try to find position/s where one or more patterns (also called strings) are occurred in text. In this study, we compare 31 different pattern matching algorithms in web documents. In web documents, searching is crucial process for content extraction process. Therefore, lengths of html tags are examined for determining which algorithm or algorithms are suitable for matching process. Our experiments show that Skip Search algorithm is the best pattern matching algorithm with 0.170 ms for web documents. Moreover, it has 0.002 ms in preprocessing time and 0.168 ms in searching time.

Keywords: keywords, keywords, keywords, keywords, keywords, keywords, keywords.

INTRODUCTION

Pattern matching is defined as searching through a string of characters looking for instances of a given "pattern" string; we wish to find all positions [1]. Pattern matching is used for tasks such as intrusion detection, virus scanning, and information retrieval. The well-known Knuth-Morris-Pratt (KMP) [1] and Boyer–Moore (BM) algorithms [2] were created to search for single patterns, while the Aho-Corasick (AC) [3] and Wu-Manber (WM) [4] multi-pattern matching algorithms are capable of inspecting multiple pattern sets simultaneously [5].

HTML (Hyper Text Markup Language) is a markup language used to create web pages on the internet. HTTP (Hyper Text Transfer Protocol) is used to transfer HTML files. HTML files are stored on the server computer's hard disk.

In this paper we have performed pattern matching tests on HTML files. 31 different pattern matching algorithms have been tested and performance results have been given in following sections. We focused on matching single patterns in our trials. When we viewed the previous works of pattern matching, we found that there were no studies in HTML files.

PATTERN MATCHING ALGORTHMS

There are many studies in the literature based on pattern matching algorithms. Pattern matching algorithms are used in researches of many fields like biologic sequences, information retrieval, image processing etc. Some well-known pattern matching algorithms are described below.

Horspool Algorithm:

The bad-character shift used in the Boyer-Moore algorithm [2] is not very efficient for small alphabets, but when the alphabet is large compared with the length of the pattern, as it is often the case with the ASCII table and ordinary searches made under a text editor, it becomes very useful.

Using it alone produces a very efficient algorithm in practice. Horspool proposed to use only the bad-character shift of the rightmost character of the window to compute the shifts in the Boyer-Moore algorithm. [6]

Quick Search Algorithm:

The Quick Search algorithm uses only the bad-character shift table [2]. After an attempt where the window is positioned on the text factor y[j .. j+m-1], the length of the shift is at least equal to one. So, the character y[j+m] is necessarily involved in the next attempt, and thus can be used for the bad-character shift of the current attempt [7].

Skip Search Algorithm:

For each character of the alphabet, a bucket collects all the positions of that character in x. When a character occurs k times in the pattern, there are k corresponding positions in the bucket of the character. When the word is much shorter than the alphabet, many buckets are empty. [8]

Simon Algorithm:

Simon indicates that the underlying automaton of Knuth-Moris-Pratt algorithm can be completed in an efficient way [9].

All algorithms are described in a web page http://www-igm.univ-mlv.fr/~lecroq/string/. By using single pattern matching manner, we have tested 31 different pattern matching algorithms which are given in Table 1.

Table 1. *Algorithms used in this study*

Apostolico Crochemore
Apostolico Giancarlo
Backward Nondeterministic Dawg Matching
Backward Oracle Matching
Boyer Moore
Brute Force
Colussi
Deterministic Finite Automaton
Forward Dawg Matching
Galil Giancarlo
Horspool
Karp Rabin
KMP Skip Search
Knuth Morris Pratt
Maximal Shift
Morris Pratt
Not So Naive
Optimal Mismatch
Quick Search
Raita
Reverse Colussi
Reverse Factor
Shift Or
Simon
Skip Search
Smith
String Matching on Ordered Alphabets
Tuned Boyer Moore
Turbo BM
Turbo Reverse Factor
Zhu Takaoka

Table 2. *Information about dataset*

	Size of Files (KB)	**Pattern Size (Character Size)**
Average	142.8	33.9
Minimum	21.1	18
Maximum	905.9	127

EXPERIMENTS

5 web pages for every 20 different online newspapers such as Milliyet, Sabah, Pravda, Corriere, Dbalears, Dnevnik, JapanTimes etc. are used in the experiments. At least two patterns have been prepared for each domain. Then, these patterns are tested five times for the relevant domains in 31 different algorithms. Table 2 gives information about file sizes of web pages and character sizes of rules in dataset.

Tags with different lengths are searched in files and each file has different size. All durations are noted to calculate average durations for each algorithm. Besides this average lengths of tags are shown in Table 3 where pattern matching results are take place. We have parsed 5 web pages in 20 domains separately. So we have used 100 HTML files to obtain results. In each file we have matched 44 patterns.

In addition to this, preprocessing and searching times are taken in each file for each algorithm. We also give the average results of preprocessing and searching times for algorithms in Table 3. It is obvious that the pattern matching processes of the algorithms which have no preprocessing phase are same with the searching time. On the other hand, the algorithms which have preprocessing phase have duration values as sum of searching and preprocessing phases durations.

We carried out our experiments on an Intel Core i5-3.2Ghz 8 GB RAM computer with Windows 10 operating system. We have developed the test software on .NET 4.5.2 framework C# Programming Language.

The preprocessing of an algorithm consists of collecting some information like statistical data or character ordering about pattern and making a model to search for.

In our experiments, best pattern matching score belongs to Skip Search algorithm. Also this algorithm has little preprocessing with 0.002 ms and the searching time with 0.168 ms. Although the Horspool algorithm has a little poor preprocessing results with 0.222 ms, the best algorithm is in searching with 0.081 ms. In the Horspool algorithm, once the document is prepared for searching, it can be searched many times quickly.

Karp Rabin, Morris Pratt, Apostolico Crochemore and Knuth Morris Pratt are little preprocessing time but their searching results are average. In total results, Tuned Boyer Moore, Horspool, Backward Nondeterministic Dawg Matching, Optimal Mismatch, Raita, Boyer Moore, Maximal Shift, KMP Skip Search, Quick Search, Turbo BM and Backward Oracle Matching algorithms have very close duration values.

BruteForce, String Matching on Ordered Alphabets and Not So Naïve algorithms are not preprocessing time but their searching results are 0.676, 1.079 and 0.522, respectively. The top results of best 12 algorithms are given in the Fig. 1.

Table 3. *Algorithms used in this study*

Algorithm	PreProcessing Time	Search Time	Total
Apostolico Crochemore	0.001	0.973	0.974
Apostolico Giancarlo	0.225	0.935	1.160
Backward Nondeterministic Dawg Matching	0.193	0.111	0.304
Backward Oracle Matching	0.005	0.384	0.390
Boyer Moore	0.226	0.102	0.328
Brute Force	0.000	0.676	0.676
Colussi	0.003	1.521	1.525
Deterministic Finite Automaton	26.402	2.846	29.248
Forward Dawg Matching	34.381	7.546	41.928
Galil Giancarlo	0.004	1.683	1.687
Horspool	0.222	0.081	0.302
Karp Rabin	0.001	1.168	1.169
KMP Skip Search	0.190	0.175	0.365
Knuth Morris Pratt	0.001	0.979	0.981
Maximal Shift	0.239	0.093	0.332
Morris Pratt	0.001	0.977	0.978
Not So Naive	0.000	0.521	0.522
Optimal Mismatch	0.228	0.091	0.319
Quick Search	0.221	0.149	0.370
Raita	0.239	0.083	0.322
Reverse Colussi	32.005	0.115	32.121
Reverse Factor	35.864	0.279	36.143
Shift Or	0.271	0.717	0.988
Simon	0.002	1.434	1.436
Skip Search	0.002	0.168	0.170
Smith	0.426	0.161	0.587
String Matchingon Ordered Alphabets	0.000	1.079	1.079
Tuned Boyer Moore	0.215	0.082	0.297
Turbo BM	0.223	0.148	0.371
Turbo Reverse Factor	34.604	0.334	34.938
Two Way	0.002	0.796	0.798

CONCLUSION

Pattern matching on strings is an important subject which has been studied several times. Many algorithms have been developed to search for patterns through the strings. In this paper, we want to observe the performances of pattern matching algorithms on HTML files. A lot of trials have been done to have meaningful results. Experimental results show that different pattern matching algorithms can retrieve data from web pages for different purposes. It would be useful to use this feature in retrieving only demanded data from web pages. All codes are an open-source and available via the github:

https://github.com/erdincuzun/SMA.NET.

ACKNOWLEDGEMENTS

The authors acknowledge the support received from the Namık Kemal University Research Fund.

REFERENCE

[1] Knuth D.E., Morris J.H., Pratt V.R., “Fast Pattern Matching in Strings”, SIAM Journal on Computing, Volume 6, Issue 2, pp. 323–350

[2] Boyer R.S.; Moore J.S., “A fast string searching algorithm”. Commun. ACM 1977, 20, pp . 762–772.

[3] Aho A.V.; Corasick M.J. “Efficient string matching”, An aid to bibliographic search. Commun. ACM 1975, 18, pp. 333–340.

[4] Manber Wu, S.; Manber, U. “A Fast Algorithm for Multi-Pattern Searching”; Technical Report TR-94-17; Department of Computer Science, University of Arizona: Tucson, AZ, USA, 1994

[6] Horspool R.N., “Practical fast searching in strings”, Software - Practice & Experience, 1980,10(6), pp. 501-506.

[7] Sunday D.M., “A very fast substring search algorithm”, Communications of the ACM. 1990, 33(8), pp.132-142

[8] Charras C., Lecroq T., Pehousek J.D., “A very fast string matching algorithm for small alphabets and long patterns”, in Proceedings of the 9th Annual Symposium on Combinatorial Pattern Matching, M. Farach-Colton ed., Piscataway, New Jersey, Lecture Notes in Computer Science 1448, 1998, pp 55-64, Springer-Verlag, Berlin.

[5] Lee C.L., Yang T.H., “A Flexible Pattern-Matching Algorithm for Network Intrusion Detection Systems Using Multi-Core Processors”, Algorithms 2017, 10(2), 58

[9] Simon, I., 1994, String matching algorithms and automata, in Results and Trends in Theoretical Computer Science, Graz, Austria, Karhumäki, Maurer and Rozenberg ed., pp 386-395, Lecture Notes in Computer Science 814, Springer Verlag.

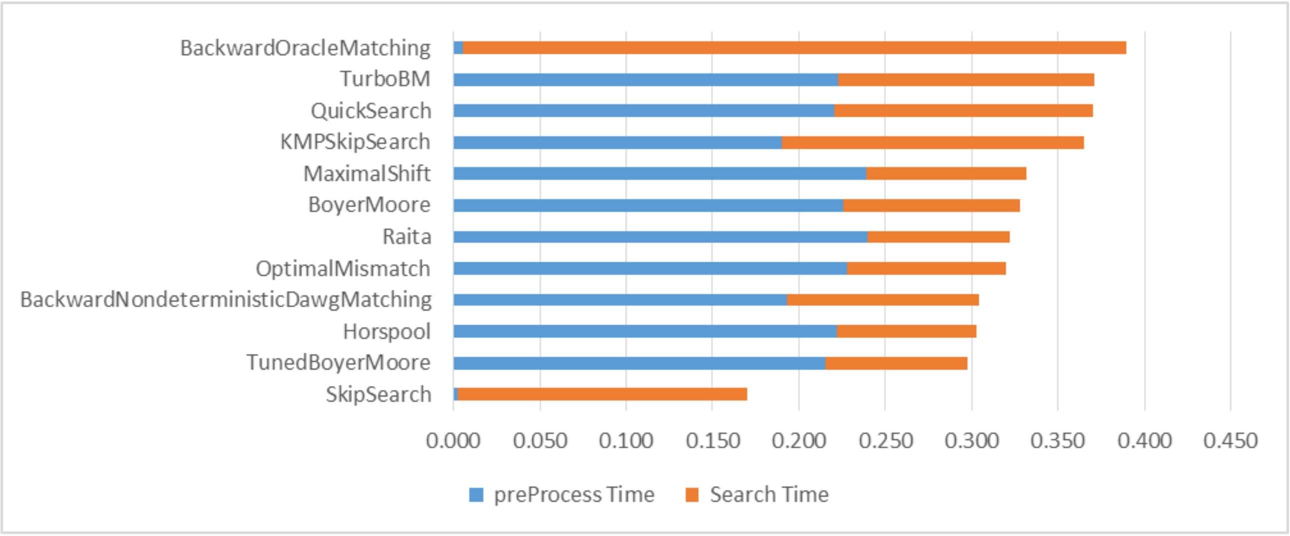

Fig. 1. *The top results of best 12 algorithms*