Global Journal of Computer Science and Technology: D
Neural & Artificial Intelligence
Volume 23 Issue 3 Version 1.0 Year 2023
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Aho-Corasick Trees for Efficient Data Storage and Machine Learning

By Mirzakhmet Syzdykov
Satbayev University

Abstract- In this work we present to reader the novel research on account for efficiency of compression algorithms like Lempel-Ziv Welch and Aho-Corasick trees. We use them to build the proper storage which is called file system in a separate or generalized stream of data. These streams weren't adopted before for big data to be compressed and queried at a fast pace. We will show further that this is the most efficient model for storing arrays of data on a server end for a final file system. The efficient algorithm for Machine Learning on Aho-Corasick trees is also presented which performs the query in linear time without getting more time on the models like neural networks which are very hardware demanding nowadays. The data structure like trie by Turing Award winner Alfred V. Aho and Margaret J. Corasick remain of big potential in the present time and are subjected to extensive research in this work.

Keywords: *trie, compression, storage, machine learning.*

GJCST-D Classification: *FoR Code: 0804*

Strictly as per the compliance and regulations of:

Aho-Corasick Trees for Efficient Data Storage and Machine Learning

Mirzakhmet Syzdykov

Abstract- In this work we present to reader the novel research on account for efficiency of compression algorithms like Lempel-Ziv Welch and Aho-Corasick trees. We use them to build the proper storage which is called file system in a separate or generalized stream of data. These streams weren't adopted before for big data to be compressed and queried at a fast pace. We will show further that this is the most efficient model for storing arrays of data on a server end for a final file system. The efficient algorithm for Machine Learning on Aho-Corasick trees is also presented which performs the query in linear time without getting more time on the models like neural networks which are very hardware demanding nowadays. The data structure like trie by Turing Award winner Alfred V. Aho and Margaret J. Corasick remain of big potential in the present time and are subjected to extensive research in this work.

Keywords: trie, compression, storage, machine learning.

I. Introduction

The algorithm dated back to 1975 by Aho and Corasick was first proposed in [1]. The matching algorithm which is linear in performance was introduced in [2, 3] where the first case is about an Aho-Corasick data stream. The work by other authors [4] focuses mainly on memory performance within the multiple patterns matching which as we know can be done in parallel.

The simulation of Aho-Corasick machine is presented in [5]. This work is important to learn as it was first to use multiple pattern search – in this work the parallel algorithm is also presented where the search is adopted as a single thread.

Nathan et al. [6] give the application of the memory efficient algorithms for string searching on Aho-Corasick trees or simply tries.

The efficiency of the performance of Machine Learning (ML) queries is first discussed in [7] where it's given within the BigData.

MXNet [8] is another ML library based upon the neural network algorithms, the performance graphs are also given in the article.

The comparison of ML libraries is presented in [9] and as we can see the efficiency still doesn't converge to the final value as the most of the programs are experimental.

Summarizing all the above we can conclude that efficient memory storages like file systems and Automated Machine Learning (AML) become more demanding nowadays, thus, we have to implement more efficient algorithms for Machine Learning Querying (MLQ) and Compressed File System (CFS) in which the Lempel-Ziv Welch (LZW) automaton is realized within the Aho-Corasick tree method of compensation of the repeated occurrences of the pattern.

We will also state the main theorems regarding the MLQ and CFS which simplifies the definition of the final complexity of the usage of these engines as software packages.

As we have stated before the repeated occurrence of pattern, which can be dualized as a single or multiple incoming query to system, should be of the minimal possible complexity from O(1) to O(n) where O(n) stands for the complexity per the query of n-words, thus giving same O(1) complexity per each of them. This result presented in this work gives the possibility of the assumption of the upper bound of performance for MLQ and CFS or, at least, such definitions are to be definitely made – we will show it and give the proofs in our next sections.

The assumption of CFS as a modified Aho-Corasick tree is a classical approach of giving its definition to broad audience which consist of not only researchers, but developers also as Application Programming Interface (API) for these engines becomes more available due to the free license.

II. Aho-Corasick Tree

These trees or simply tries are the recursive data structures which represent the deterministic finite automata (DFA) for the operational logics of querying them in a separate or parallel mode where the update operation is also supported.

Namely the tree T for arbitrary language L (A) in alphabet A is given by the recursive function f(s) for the query string s, which can be defined as follows:

$$T =< A, f: f(s) = 1 \Leftrightarrow s \in L(A) > \qquad (1)$$

As we know the language can be regular and can be given by the specially adopted operands as in regular expression or even with the extended operators like intersection, complement and subtraction.

The update of the tree is the building of a same singleton within the new string s where function f(s) becomes terminal as if we would demonstrate the work of the same compound DFA. Thus, as we have

Author: Satbayev University, Almaty, Kazakhstan.
e-mail: mspmail598@gmail.com
ORCID ID: https://orcid.org/0000-0002-8086-775X

shown that regular tree is a simple DFA, we can conclude that the operational logic is the same, to be more precise the query can be handled in O(1) per each symbol in the alphabet A for the string s. The update operation has the same complexity as we have to find the terminal symbol in a tree in the same time.

As we have defined the data structure for pre-defined operations, we can proceed to the applications of Aho-Corasick trees for efficient computations.

III. Lempel-Ziv Welch Tree

These types of trees are sub-domains of Aho-Corasick trees as they were first adopted for the compressed streams where update operation is necessary for the purpose of building the efficient Aho-Corasick tree in compression algorithms. These algorithms provide the logic of reassembling it in the final output stream within the first appearance of the character which doesn't belong to the tree, later this character is added to the tree and all the necessary output is done like the pair of output index of the state and the character value itself. This can be better generalized by the following relation:

$$T =< A, f: f(s,c) = 1 \Leftrightarrow s \in L(A), (s,c)\ not\ in\ L(A) >. \quad (2)$$

Thus, we have demonstrated that LZW trees are the abstracts of the Aho-Corasick tree and have the same performance complexity of O (1). We will demonstrate further that this fact is the most important to use within the efficient data storage like file system for big amount of data which are able of compensating both querying and update operations.

The file system built upon the LZW and Aho-Corasick trees makes the output in the global tree which is localized in the modern data compression software. Thus, we extend our tree with the new set like mark, which defines, whether this is output and of what segment of the file:

$$T =< \cdots, (file, index) \in F >. \quad (3)$$

The mark file and index in (3) gives the precise location of the tree segment, thus allowing the multiple files to be stored on the same LZW tree which as it was proved before is an extended version of Aho-Corasick tree. We define this tree globally as the total entropy of the data becomes lower when more patterns occur or they occur in a specific case.

From all the above we can give further directions of applying not only LZW trees but the DFA built upon the arbitrary regular expression which takes place in the formation of the randomly categorized output format like, for example, Portable Executable (PE).

IV. Efficient Machine Learning

In this section we demonstrate the application of Aho-Corasick trees on the Machine Learning querying and training which plays the role of update operation in a tree.

Typically, the ML model can be defined as:

$$M =< T, F, O >. \quad (4)$$

where M is a model, T is a set of terminal rows, F is a set of weighted values and O is a set of result actions upon querying.

The expression (4) can be composed as the limited number of rows in a file as it's realized in Microsoft ML .NET package. The terminals T are the parts of query as well as the set of values F where T is the entities and F are the weights against which this query is to be tested within the given threshold. The set of outputs O is the set of non-terminal symbols which are to be queried in case of successful matching on a model M.

We use the Aho-Corasick tree for terminals T in our efficient and parallel algorithm and a threshold t for the arbitrary query q:

$$q.T = T, |F - q.F| \leq t \Rightarrow yield\ O. \quad (5)$$

Thus, from the expression (5) it follows that Aho-Corasick tree can be successfully built upon the given input and updated as well. The query operation is given on a tree within the defined threshold and the size of the model will be physically given in O(n) where n is the total size of this model.

In order to build the model we have to give its total height as O (|T| + |F|) – within these bounds it can be seen more practical for the business development of MLQ.

The main theorem which we state and prove is that the size of the model as well as the number of steps required for the querying and update of the arbitrary structured data like set of patterns or ML model is linear within the actual size. This follows from the main proofs to the Aho-Corasick trees.

V. Conclusion

We have demonstrated the power of data structure like tries or Aho-Corasick trees within the implementation of fully functional and efficient data storage based on the file system. Our research was also focused on the efficiency of non-neural models of operation for effective ML. We have obtained linear results which are most efficient to the present day.

Our main goal was to develop the unary data structures and the main ideas are given in the sections of this work. The results can be addressed to the Big Data where the performance plays vital role and, thus,

is to be of minimal possible value of O (n), where n is the number of input data.

For this purpose, we give non-neural and non-consumptive solutions for both data storage and the ML usage in systems with big amount of data and where the parallel computations are made as per demand of the users of the system.

Acknowledgements

The author expresses gratitude to the A. M. Turing Award winner Alfred V. Aho and Margaret J. Corasick for the implementation of the most efficient data structure for multiple queries and updates as well as to all the Association for Computing Machinery (ACM) for advancement in an algorithmic field.

Funding

This work was fully supported by an educational grant of MES RK.

References Références Referencias

1. Aho, Alfred V., and Margaret J. Corasick. "Efficient string matching: an aid to bibliographic search." *Communications of the ACM* 18.6 (1975): 333-340.
2. Tao, Tao, and Amar Mukherjee. "Multiple-Pattern Matching In LZW Compressed Files Using Aho-Corasick Algorithm." *DCC*. 2005.
3. Aldwairi, Monther, Abdulmughni Y. Hamzah, and Moath Jarrah. "MultiPLZW: A novel multiple pattern matching search in LZW-compressed data." *Computer Communications* 145 (2019): 126-136.
4. Liangxu, Sun, and Li Linlin. "Improve Aho-Corasick algorithm for multiple patterns matching memory efficiency optimization." *J. Convergence Inf. Technol. (JCIT)* 7 (2012): 19.
5. Kida, Takuya, et al. "Multiple pattern matching in LZW compressed text." *Proceedings DCC'98 Data Compression Conference (Cat. No. 98TB100225). IEEE*, 1998.
6. Tuck, Nathan, et al. "Deterministic memory-efficient string matching algorithms for intrusion detection." *IEEE INFOCOM 2004*. Vol. 4. IEEE, 2004.
7. Al-Jarrah, Omar Y., et al. "Efficient machine learning for big data: A review." *Big Data Research* 2.3 (2015): 87-93.
8. Chen, Tianqi, et al. "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems." *arXiv preprint arXiv*:1512.01274 (2015).
9. Nguyen, Hoang, et al. "Efficient machine learning models for prediction of concrete strengths." *Construction and Building Materials* 266 (2021): 120950.