Online Suffix Tree Construction for Streaming Sequences

Giyasettin Ozcan and Adil Alpkocak

Dokuz Eylul University, Department of Computer Engineering,
Tinaztepe Buca 35160, Izmir, Turkey
giyaseddin.ozcan@deu.edu.tr, alpkocak@cs.deu.edu.tr

Abstract. In this study, we present an online suffix tree construction approach where multiple sequences are indexed by a single suffix tree. Due to the poor memory locality and high space consumption, online suffix tree construction on disk is a striving process. Even more, performance of the construction suffers when alphabet size is large. In order to overcome these difficulties, first, we present a space efficient node representation approach to be used in Ukkonen suffix tree construction algorithm. Next, we show that performance can be increased through incorporating semantic knowledge such as utilizing the frequently used letters of an alphabet. In particular, we estimate the frequently accessed nodes of the tree and introduce a sequence insertion strategy into the tree. As a result, we can speed up accessing to the frequently accessed nodes. Finally, we analyze the contribution of buffering strategies and page sizes on performance and perform detailed tests. We run a series of experimentation under various buffering strategies and page sizes. Experimental results showed that our approach outperforms existing ones.

Keywords: Suffix trees, sequence databases, time series indexing, poor memory locality.

1 Introduction

Suffix trees, are versatile data structures which enable fast pattern search on large sequences. The large sequence, data set, can be a DNA sequence of a human, whose length is 3 billion; or it can be a collection of musical fragments, where number of fragments in the collection is large but average length of each fragment is moderate[15]. In both sequence cases, total size of the data set may be extremely large. For such large sequence sets, suffix trees introduce a fundamental advantage; sequence search time does not depend on the length of the data set.

The concept of suffix tree construction was initiated before the seventies by a brute force approach [11]. For each suffix insertion, brute force approach preceded a common prefix search operation in the tree. Nevertheless, it was not practical since computational cost of the brute force suffix tree construction was at least exponential. In the seventies, linear time suffix tree construction algorithms were introduced using suffix links and tested on memory. [18,28]. In these algorithms, suffix links functioned as shortcuts, which enable fast access to the suffix insertion positions of the tree. In other words, they hold the address of a node, which contributes to the next suffix insertion position. As a result, traversing the tree for each suffix insertion was

H. Sarbazi-Azad et al. (Eds.): CSICC 2008, CCIS 6, pp. 69–81, 2008.

not necessary; instead suffix links from the previous step supplied the direct address. Due to this strong advantage, linear time suffix tree construction became possible. [11].

Although early suffix tree construction algorithms ensure linear time construction, they share a common pitfall: the offline property. For instance, in McCreight algorithm [18], all letters of the sequence should be scanned before suffix tree construction procedure starts up. Such situation may cause an important constraint, if, for example, the occurrence of the rightmost letter is delayed. Twenty years after McCreight, Ukkonen has presented an online version [27]. In the online construction algorithm, the scanned part of the sequence can be projected to the suffix tree whereas; it is possible to extend the suffix tree by reading the next letter from the sequence.

Advancement on the suffix tree construction took a step by Generalized Suffix Tree (GST) [3]. Bieganski looked at the problem from a different aspect and pointed out the importance of indexing multiple sequences in a single suffix tree. In GST, it was necessary to identify the origin of each sequence. Hence extra node identifiers were added within leaf nodes. As a result of GST, most of the symbolic representations of Time Series could be indexed by a single suffix tree [12].

Suffix tree construction on disk leads to important difficulties such as high space consumption and poor memory locality. Concretely, space consumption of a suffix tree node is high and fewer nodes can fit into a page. If a suffix tree contains large number of nodes, disk page requirement of a suffix tree will be large as well. On the other hand, poor memory locality is inevitable since suffix tree nodes are generated in random order and nodes of a selected path are generally spread across different pages. Because of this, traversal on a path frequently leads to indispensable page misses.

Recently, some researchers pointed out disk based suffix tree algorithms. In 1997, Farach-Colton proposed a theoretical algorithm, which ensured linear time construction, [7] but his algorithm has not supported by practical results. In PJama platform, authors suggested a space efficient algorithm by removing suffix links from suffix tree [13]. In addition they grouped the suffixes of a text according to their common prefixes. Therefore, suffixes in the same group can be inserted into the tree one by one. Hence both poor memory locality and space consumption drawbacks would be improved. Recently, new studies have followed a similar path [4, 22, 26, 29]. Nevertheless, these algorithms did not consider online property of suffix trees. and put constraints on dynamic sequence insertions. Although [23] introduces an online algorithm, it is not incremental and put constraints on streaming sequence insertions. In fact all these algorithms were designed for large genomic data sets and do not consider medium length streaming sequences such as MIDI.

In 2004, Bedathur and Haritsa presented an online suffix tree construction algorithm on disk [2] Based on Ukkonen's strategy, they considered physical node representations on a tree. In addition, they introduced a buffering strategy. Nonetheless, they did not test medium-size streaming sequences.

In this study, we present an Online Generalized Suffix Tree (OGST) approach on disk. To the best of our knowledge, this is the first study dealing with OGST construction on secondary memory. Briefly, contribution of this study is threefold: First, we modify the suffix node trees so that direct access to parent becomes possible. Second,