Pattern Recognition Letters 116 (2018) 97–100

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Applications of single-operator edit distances for permuted sequences

Jeffrey Uhlmann

Department of Electrical Engineering and Computer Science, University of Missouri, Columbia, MO 65211, USA

ARTICLE INFO

Article history:
Received 6 October 2017
Available online 21 September 2018

MSC:
41A05
41A10
65D05
65D17

Keywords:
Edit distance
Permutations
Data compression

ABSTRACT

In this paper we consider single-operator distance functions for comparing sequences/strings that are equivalent up to permutation. It is shown that these distance functions admit smaller backtrace expressions than conventional edit distance. Results are provided showing that this reduced backtrace complexity can be exploited for data compression and post-analysis modeling of packet delays in a communications network application.

1. Introduction

Edit distance and its specialized variants, e.g., Levenshtein, are fundamental tools for pattern recognition because they can reveal structural similarities between complex objects [8]. They are applied widely for the comparison of textual and genomic sequences [10,17] and for complex tree and graph structures [2,13,14]. Closely related to the calculation of edit distance is the generation of a *backtrace*, i.e., a minimal sequence of edit operations to transform one sequence to another. This sequence of operations can be regarded as a compressed representation of the second sequence given the first sequence. In a genomic application, for example, a "typical" baseline sequence could be identified/defined so that all related sequences could then be represented in compressed form as a sequence of edit operations applied with respect to the baseline sequence.

As an example, suppose we are given two sequences. The first consists of n distinct symbols, i.e., it is a permutation, and the second sequence is identical to the first except that the last two symbols are interchanged (swapped). The two sequences can be stored in compressed form by writing out the first sequence verbatim followed by an instruction saying that the second sequence can be obtained from the first one just by swapping the last two symbols. This reduces the space requirement for storing the two sequences by almost 50% for large n, and this level of compression cannot be achieved using standard compression algorithms, e.g., Huffman encoding, which assume that some symbols appear more frequently than others.

In this paper we consider the application of edit distance information to analyze and compress permutation-equivalent sequences, i.e., sequences representing orderings of the same set of distinct symbols. As will be shown, this assumption can provide improved backtrace compression by allowing the effective number of primitive edit operations to be reduced in certain practical applications.

2. Backtrace complexity

Conventional string edit distance is defined in terms of three primitive edit operations: *insertion, deletion*, and *substitution*. The third operation, substitution, can be expressed as a deletion followed by an insertion, so it is redundant if its weighted cost equals the sum of the costs of deletion and insertion. Assuming unit cost for each operation, the *backtrace* for strings s_1 and s_2, $d(s_1, s_2) = k$, is a sequence of k operations that transforms s_1 to s_2.

In the earlier example involving a sequence s_1 of n distinct symbols, with another sequence s_2 that is equivalent to s_1 except that its last two symbols are swapped, the edit distance between the two sequences is 2. The backtrace obtained from calculating this edit distance must give two edit operations that transform s_1 to s_2. In this case the two operations could be a deletion followed by an insertion. More specifically, deleting the last symbol of s_1, and then inserting that symbol before its predecessor, has the effect of swapping the last two symbols of s_1 to give s_2. The backtrace is not generally unique, and in this case the same result could be achieved with two subsitutions.

E-mail address: uhlmannj@missouri.edu

https://doi.org/10.1016/j.patrec.2018.09.021
0167-8655/

If s_1 is written out explicitly, followed by some kind of representation of the two edit operations sufficient to obtain s_2, the amount of space saved depends on the length (number of bits) needed to represent the backtrace, i.e., to specify the two edit operations. For general edit distance there are 3 edit operations (insertion, deletion, and substitution) and they can be distinguished using 2 bits because $\lceil \log_2(3) \rceil = 2$. For example, 00 could represent *Insert*, 01 could represent *Delete*, and 10 could represent *Substitute*. It then remains to determine how many bits are needed to identify their respective arguments, e.g., to specify the symbol and location for an insertion operation.

If edit distance is applied to two strings s_1 and s_2 of length n, the number of bits required to represent each operation in a backtrace can be explicitly calculated:

1. An operator ID is required to uniquely identify which of the three edit operations is to be performed: 2 bits.
2. Deletion requires a single argument giving the index of the symbol to be deleted: $\lceil \log_2(n) \rceil$ bits.
3. Substitution requires two arguments giving, respectively, the index of the character to be replaced and the symbol to replace it: $2 \times \lceil \log_2(n) \rceil$ bits.
4. Insertion requires two arguments giving, respectively, the symbol to insert and the index of a location (which may include the position *before* the first symbol of the sequence or *after* the last symbol of the sequence) at which the symbol is to be inserted: $\lceil \log_2(n) \rceil + \lceil \log_2(n+1) \rceil$ bits.

In addition to the number of bits required for each of the k operations, a preamble (header) of $\lceil \log_2(n) \rceil$ bits are required to specify the total number of operations that follow, i.e., the edit distance.

Assuming that all three operations appear with the same frequency, the expected total number of bits can be slightly overestimated as:

$$\underset{\#\text{ of operations}}{\lceil \log_2(n) \rceil} + \underset{\text{op ID}}{2k} + \underset{\text{avg argument length}}{2k \cdot \lceil \log_2(n) \rceil} \tag{1}$$

where the overestimate derives from assuming that an extra bit may be required for the location index for each insert operation (which leads to rounding of the coefficient on the expected argument-length term from 5/3 to 2).

Given that there are typically many equivalent backtraces, i.e., many different sequences of k operations, there would seem to be an advantage to choosing the one that uses the most deletions, since a *Delete* requires fewer bits because it has only one argument. In the case of permutation-equivalent strings, however, symbols are neither created nor eliminated so deletion must have a corresponding insertion or substitution, thus a lower relative cost for deletion cannot translate to reduced backtrace size. More generally, it can be observed that combinations of insertions, deletions, and substitutions can essentially only result in the moving of symbols. This motivates the following definition of a primitive edit operation:

> **Definition**: A *move* operation applied to a single symbol in a sequence is defined to be a relocation of the symbol from one index position to another index position, i.e., equivalent to deletion of that symbol from the sequence followed by its insertion at a different location in the sequence.

Given permutation-equivalent sequences s_1 and s_2, a move-based edit distance between them can be defined as the number of symbol moves required to transform s_1 into s_2. Such an edit distance based only on move operations is sometimes referred to as *permutation distance* or *Ulam distance* [1].

It is reasonable to expect permutation distance based only on move operations to be exactly half that of ordinary edit distance restricted to unit-cost insertion and deletion operations. While this is true, the backtrace size is significantly reduced in the case of single-move operations because there is no overhead required to distinguish between insert and delete operators. This leads to a precise estimate of its backtrace bit requirement[1]:

$$\lceil \log_2(n) \rceil \; + \; 2k \cdot \lceil \log_2(n) \rceil \tag{2}$$

which represents a savings of almost $2k$ bits compared to standard edit distance due to elimination of 2-bit operation designations.

A concrete example may provide a more intuitive picture of how the preceding applies to the compressed representation of permutation-equivalent sequences. Consider the following two sequences:

$$s_1: \quad \texttt{u v w x y z} \tag{3}$$

$$s_2: \quad \texttt{u x v y w z} \tag{4}$$

They can be stored explicitly as:

$$\texttt{u v w x y z u x v y w z} \tag{5}$$

or in compressed form (only slightly compressed in this case) as s_1 followed by the move operations sufficient to obtain s_2:

$$\texttt{u v w x y z 2 3 2 4 1} \tag{6}$$

where the first 2 signifies that the edit distance between s_1 and s_2 is 2, therefore two move operations will follow. The subsequent "3 2" signifies that the symbol at location 3 (indexing from 0) of s_1 is to move to position 2, and "4 1" says that the symbol at location 4 of the result should move to position 1.

Applications of the metric properties of edit distance for efficient database search [6,9,16] and clustering [7,15] have been extensively studied for more than 25 years. In the following section we demonstrate how single-operation edit distances can be used in a very different type of application to both analyze and – as a byproduct – compress data logged from a communications network for purposes of post-analysis modeling of packet delays.

3. Permuted sequence analysis and compression

Consider a network of V nodes in which each node may communicate with every other node via links that connect each node to a non-empty subset of the remaining nodes in the network. Assuming the number of links E is $O(V)$, i.e., the network can be represented as a sparse graph, the path a packet travels from node i to node j will involve a variable number of links such that the total travel time is a variable that depends both on the paths that exist between nodes i and j and also the characteristics (e.g., including environmental variables) of the links comprising those paths at the time of traversal. Mitigating the effect of such delays is a critical challenge in many distributed information processing applications [4,12].

One way to assess the delay properties of the network is to select a particular node, node 0, to send a sequence s_0 of n packets to all of the other $V-1$ nodes. This permits the temporal sequence s_i as received by node i to be compared to the original sequence s_0, with any deviations attributable to packet delays. The minimal sequence of move operations to transform the sequence received at node i to/from the sequence sent by node 0, i.e., the permutation-distance backtrace, provides an explicit characterization of the pattern of delays. As a secondary benefit it can be used to compress the collected data for subsequent analysis.

[1] It should be noted that the location argument is strictly $\lceil \log_2(n) \rceil$ (not $\lceil \log_2(n+1) \rceil$), because after a deletion the subsequent insertion is applied to a sequence of $n-1$ symbols.