Similarity of Event Sequences

Extended abstract

Heikki Mannila and Pirjo Ronkainen
University of Helsinki, Department of Computer Science
P.O. Box 26, FIN-00014 Helsinki, Finland
Heikki.Mannila@cs.Helsinki.FI, Pirjo.Ronkainen@cs.Helsinki.FI

Abstract

Sequences of events are an important form of data that occurs in many application domains, such as telecommunications, biostatistics, user interface design, etc. We present a simple model for measuring the similarity of event sequences, and show that the resulting measure of distance can be efficiently computed using a form of dynamic programming.

Keywords: *sequences of events, distance, dynamic programming*

1 Introduction

Sequences of events are a common form of data that can contain important knowledge to be discovered. Examples of such data are telecommunications network alarms, user interface actions, crimes committed by a person, occurrences of recurrent illnesses, WWW page requests, etc. Recently, interest in knowledge discovery from sequences of events has increased: see, e.g., [4, 5, 11, 8, 2, 9, 7, 6].

Similarity between objects is a fundamental notion that has to be defined before one can apply various statistical, machine learning, or data mining methods. For example, to use case-based reasoning or clustering one has to be able to determine when one sequence of events resembles another.

In this paper we consider the problem of defining similarity, or distance, between event sequences. We start with the intuitive idea that similarity between sequences should somehow reflect the amount of work that has to be done to convert one sequence to another. We formalize this notion as *edit distance* [1, 3, 10] between sequences, and show that the resulting definition of similarity has several appealing features. We apply a simple and well-known dynamic programming framework for computing the edit distances and show that the resulting algorithm is efficient in practice.

In detail, the paper is organized as follows. In Section 2 we present the model for event sequences and give some examples. Section 3 gives the definition of similarity between sequences as edit distance between two objects under a given set of transformation operations. In Section 4 we give a straightforward dynamic programming algorithm for computing the similarity measure, and in Section 5 we present some empirical results. Section 6 is a short conclusion.

2 Event sequences

A realistic way of modeling events is to consider a given set $R = \{A_1, \ldots, A_m\}$ of *event attributes* with domains $D_{A_1}, \ldots, D_{A_m}$. An *event* e over R is a $(m+1)$-tuple $(a_1, \ldots, a_m, t)$, where $a_i \in D_{A_i}$ and t is a real number, the *occurrence time* of e. An *event sequence* $\mathcal{S}$ is a collection of events over R, i.e., a relation over $R \cup \{T\}$, where the domain of attribute T is the set of real numbers.

Example 1 In the telecommunications domain, event attributes are, e.g., *type*, *module*, and *severity*, indicating the type of alarm, the module that sent the alarm, and the severity of the alarm, respectively. □

Example 2 In analyzing a WWW log, the events have attributes *page* (the accessed page), *host* (the accessing host), and time. □

In this paper, we mainly consider a simplified model of events, where each event has only a type and an occurrence time. Given a class $\mathcal{E}$ of elementary *event types*, an *event* is then a pair (e, t), where $e \in \mathcal{E}$ and t is an integer. An *event sequence* S is an ordered sequence of events, i.e.,

$$S = (e_1, t_1), (e_2, t_2), \ldots, (e_n, t_n),$$

where $e_i \in \mathcal{E}$, and $t_i \leq t_{i+1}$ for all $i = 1, \ldots, n-1$. The length of the sequence S is, therefore, $|S| = n$.

Example 3 Assume the event types are A, B, C, and D. Two examples of sequences of events are

$$S = (D,8)\ (D,12)\ (A,15)\ (B,17)\ (A,20), \text{ and}$$
$$T = (D,4)\ (D,8)\ (C,15)\ (B,17)\ (A,18)\ (C,20). \quad \square$$

Example 4 An example sequence of a telecommunication alarm data is a sequence of (alarm type, time) -pairs

$$(2912,11)\ (7406,15)\ (7705,18)\ (2567,20)$$
$$(7705,20)\ (7210,21)\ (7705,21) \quad \square$$

The basic problem we consider in this paper is how one should define a concept of similarity or distance between event sequences. Such a notion is needed in virtually any database or knowledge discovery application on sequences. Namely, if one cannot say when two sequences are close to each other, the possibilities for making intelligent search or retrieval is quite limited.

Ideally, such distance notion $d(S,T)$ should be a metric, that is, satisfy the following requirements. For all the sequences S, T, and U we should have

$$\begin{aligned} &d(S,T) \geq 0, \\ &d(S,T) = 0 \text{ if and only if } S = T, \\ &d(S,T) = d(T,S), \text{ and} \\ &d(S,T) + d(T,U) \geq d(S,U). \end{aligned}$$

Additionally, d should be efficiently computable, and, of course, it should be in some sense natural. In the next section we show that such a notion can, in fact, be defined quite easily.

3 Similarity measures

The intuitive idea behind our definition of similarity, or distance, between event sequences is that it should somehow reflect the amount of work needed to transform a sequence to another. The definition of similarity is formalized as edit distance $d(S,T)$.

Operations. For counting the edit distance we need to define *a set of transformation operations.* We have chosen to use three natural operations:

1. $\texttt{Ins}(e,t)$ that inserts an event of the type e to time t, i.e., adds a pair (e,t) to the sequence.
2. $\texttt{Del}(e,t)$ that deletes an event of the type e from time t, i.e., deletes a pair (e,t) from the sequence.
3. $\texttt{Move}(e,t,t')$ that moves an existing event (e,t) from time t to time t'.

Costs of operations. To each operation o we associate an cost $c(o)$. The cost of an insertion operation is defined by

$$c(\texttt{Ins}(e,t)) = w(e),$$

where $w(e)$ is a constant value proportional to $occ(e)^{-1}$, where $occ(e)$ is the number of occurrences of a e-type event in a long reference sequence. With this definition the cost of adding a rare event into the sequence is higher than the cost of adding a common event. The cost of a deletion operation is defined to be the same as the cost of an $\texttt{Ins}$-operation, i.e., $c(\texttt{Del}(e,t)) = w(e)$.

The cost of a $\texttt{Move}$-operation is defined as

$$c(\texttt{Move}(e,t,t')) = V \cdot |\, t - t' \,|,$$

where V is a constant and $|\, t - t' \,|$ the length of the move. With this definition a short move has a lower cost than a long move. This definition also assumes that the occurrence times of events have approximately the same magnitude in both compared sequences. Without this assumption, sequences such as $(A,1)$ $(B,2)$ and $(A,101)$ $(B,102)$ would be considered to be far from each other. In our applications the event times typically start from zero and have approximately the same range, so the assumption holds.

Definition of distance of sequences. If the cost of an operation o_i is $c(o_i)$, and k is the number of operations in the sequence O_j, the cost of an operation sequence $O_j = o_1, o_2, ..., o_k$ is

$$c(O_j) = \sum_{i=1}^{k} c(o_i).$$

The distance $d(S,T)$ is defined as the sum of costs of the cheapest sequence of operations transforming the sequence S to the sequence T. That is,

$$d(S,T) \;=\; min\{\ c(O_j) \mid O_j \text{ is an operation sequence transforming a sequence } S \text{ to a sequence } T\ \}.$$

The following proposition follows directly from the definitions.

Proposition 5 The distance d defined above is a metric. $\square$

Parameters. In some applications it makes sense to limit the length of moves. This bound W can be some predefined value, *window size*, in given time units. It can also be defined as the length of the time period between the occurrence of the first and the last event of the longer of the event sequences compared. Consequently, the cost of a $\texttt{Move}$-operation is always $cost(\texttt{Move}) \leq V \cdot W$.

The parameter V has some logical restrictions. For all event types e we should have $V \leq 2 \cdot w(e)$. Namely, if $V > 2 \cdot w(e)$, it is never useful to move an event e: one can always delete and insert an event, instead.

Example 6 With the definitions above, the minimum cost of transforming the sequence S to the sequence T in Example 3 would be

$$\begin{aligned} c(S \rightarrow T) &= c(\mathtt{Move}(D,8,4)) + c(\mathtt{Move}(D,12,8)) \\ &\quad + c(\mathtt{Ins}(C,15)) + c(\mathtt{Del}(A,15)) \\ &\quad + c(\mathtt{Move}(B,17,17)) \\ &\quad + c(\mathtt{Move}(A,20,18)) + c(\mathtt{Ins}(C,20)) \\ &= V \cdot |8-4| + V \cdot |12-8| \\ &\quad + w(C) + w(A) + V \cdot |17-17| \\ &\quad + V \cdot |20-18| + w(C) \\ &= 2 \cdot w(C) + w(A) + V \cdot 10 \end{aligned}$$

if moving an event is always preferred to inserting and deleting. □

Example 7 Let the sequence in Example 4 now be the sequence S. With the definitions above, the minimum cost of transforming it to the sequence T

$(7277, 2)$ $(2912, 3)$ $(7277, 3)$ $(7406, 10)$
$(7410, 12)$ $(2567, 14)$ $(7705, 23)$ $(7705, 23)$

would be

$$\begin{aligned} c(S \rightarrow T) &= c(\mathtt{Ins}(7277,2)) \\ &\quad + c(\mathtt{Move}(2912,11,3)) \\ &\quad + c(\mathtt{Ins}(7277,3)) \\ &\quad + c(\mathtt{Move}(7406,15,10)) \\ &\quad + c(\mathtt{Ins}(7410,12)) \\ &\quad + c(\mathtt{Del}(7705,18)) \\ &\quad + c(\mathtt{Move}(2567,20,14)) \\ &\quad + c(\mathtt{Move}(7705,20,23)) \\ &\quad + c(\mathtt{Del}(7210,21)) \\ &\quad + c(\mathtt{Move}(7705,21,23)) \\ &= w(7277) + V \cdot |11-3| \\ &\quad + w(7277) + V \cdot |15-10| \\ &\quad + w(7410) + w(7705) \\ &\quad + V \cdot |20-14| + V \cdot |20-23| \\ &\quad + w(7210) + V \cdot |21-23| \\ &= 2 \cdot w(7277) + w(7410) + w(7705) \\ &\quad + w(7210) + V \cdot 24 \end{aligned}$$

if moving an event is always preferred to inserting and deleting. □

The distance notion defined by looking at the cost of transforming one sequence to another is fairly natural. One can show that this notion generalizes some distance measures for time series, and also that it generalizes the notions of edit distance for strings. We omit the details for reasons of brevity.

The model presented above can easily be extended to allow for edit operations that change the event type. If the set $\mathcal{E}$ of event types has a metric $h_{\mathcal{E}}$ defined on it, one can, for example, define that the cost of a transformation of an event (e, t) to another event (e', t') is $h_{\mathcal{E}}(e, e') + b$, where b is a constant.

4 Dynamic programming algorithm

We use a fairly typical dynamic programming [1, 3, 10] for finding the optimal operation sequence of transforming the sequence $S = (e_1, t_1), (e_2, t_2), \ldots, (e_n, t_n)$ to the sequence $T = (f_1, u_1), (f_2, u_2), \ldots, (f_m, u_m)$. We actually compute the *weighted edit distance* of the two sequences. Given the sequences S and T, we use $r(i, j)$ to denote the minimum cost of the operations needed to transform the first i events of the sequence S into the first j events of the sequence T. With this definition, the weighted edit distance of the sequences S and T is $r(n, m)$, where n is the number of events in the sequence S and m the number of events in the sequence T.

The base conditions and the recurrence relation for the value $r(i, j)$ are

$$\begin{aligned} r(0,0) &= 0 \\ r(i,0) &= r(i-1,0) + w(e_i) \\ r(0,j) &= r(0,j-1) + w(f_j) \\ r(i,j) &= min\ \{\ r(i-1,j) + w(e_i), \\ &\qquad r(i,j-1) + w(f_j), \\ &\qquad r(i-1,j-1) + k(i,j)\ \} \end{aligned}$$

where $w(e_i)$ and $w(f_j)$ are costs of inserting (deleting) of a e_i-type or f_j-type event, respectively, and

$$k(i,j) = \begin{cases} V \cdot |t_i - u_j|, & \text{if } e_i = f_j \\ w(e_i) + w(f_j), & \text{if } e_i \neq f_j. \end{cases}$$

In more detail, the algorithm is as follows.

$r(0, 0) = 0$;
for $i = 1$ **to** m **begin**
    $r(i, 0) = r(i-1, 0) + w(e_i)$; **end**
for $j = 1$ **to** n **begin**
    $r(0, j) = r(0, j-1) + w(f_j)$; **end**
for $i = 1$ **to** m **begin**
    for $j = 1$ **to** n **begin**
        if $e_i = f_j$ **then**
            $k(i, j) = V \cdot$ the time difference between e_i and f_j;
        else
            $k(i, j) = w(e_i) + w(f_j)$;
        fi
        $r(i, j) = min\ \{\ r(i-1, j) + w(e_i)$,
            $r(i, j-1) + w(f_j)$,

$r(i-1, j-1) + k(i,j)$ };

end;

end;

It is straightforward to verify that the algorithm computes the desired distance.

5 Empirical results

All the experiments were run on a PC with 90 MHz Pentium processor and 32 MB main memory, under the Linux operating system. The sequences of (event type, time)-pairs resided in a flat text file.

We tested the naturalness of the similarity definition by using a telecommunication alarm sequence that consists of 7185 alarms covering a time period of 39 days. There are 80 different types of alarms with diverse frequencies and distributions. All the alarms in this sequence are sent by one network element. We selected from this sequence a subsequence representing the alarms occurring in a given 15 minute long interval (20 alarms). We computed the similarity of this sequence with all sequences consisting of alarms from the same length of time, and inspected the results by eye. Each comparison took only a fraction of a second. The maximally similar sequences are, indeed, intuitively fairly close to the original sequence. Obviously, the choice of the parameters for the costs of different types of operations has a strong influence on the results.

As for the efficiency of the algorithm, we tested it on random sequences with 10 event types and times varying uniformly from 1 to 100 000. The sequences were of length 100 – 500. As expected, the time needed for the comparison of two sequences increases quadratically with the length of the sequences. For sequences of length 100, the simple implementation of the algorithm took about 4.7 seconds, for sequences of length 500, approximately 108 seconds were needed. In most of the applications that we have been considering, the sequences to be tested for similarity are fairly short, so the quadratic behavior of the algorithm is not a problem.

6 Concluding remarks

We have described a simple method for defining similarity between sequences of events. We claimed that the definition produces an intuitively appropriate notion of similarity. The method can be implemented using a straightforward dynamic programming idea, and the results of preliminary experiments seem promising.

References

[1] A. V. Aho. Algorithms for finding patterns in strings. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, pages 255 – 400. Elsevier Science Publishers B.V (North-Holland), Amsterdam, The Netherlands, 1990.

[2] C. Bettini, X. S. Wang, and S. Jajodia. Testing complex temporal relationships involving multiple granularities and its application to data mining. In *Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'96)*, pages 68 – 78, Montreal, Canada, June 1996.

[3] M. Crochemore and W. Rytter. *Text Algorithms*. Oxford University Press, New York, 1994.

[4] C. Dousson, P. Gaborit, and M. Ghallab. Situation recognition: Representation and algorithms. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 166 – 172, Chambery, France, Aug. 1993.

[5] P. Laird. Identifying and using patterns in sequential data. In K. Jantke, S. Kobayashi, E. Tomita, and T. Yokomori, editors, *Algorithmic Learning Theory, 4th International Workshop*, pages 1 – 18, Berlin, 1993. Springer-Verlag.

[6] H. Mannila and H. Toivonen. Discovering generalized episodes using minimal occurrences. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96)*, pages 146 – 151, Portland, Oregon, Aug. 1996. AAAI Press.

[7] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovering frequent episodes in sequences. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95)*, pages 210 – 215, Montreal, Canada, Aug. 1995. AAAI Press.

[8] R. A. Morris, L. Khatib, and G. Ligozat. Generating scenarios from specifications of repeating events. In *Second International Workshop on Temporal Representation and Reasoning (TIME-95)*, Melbourne Beach, Florida, Apr. 1995.

[9] T. Oates and P. R. Cohen. Searching for structure in multiple streams of data. In *Proceedings of the Thirteenth International Conference on Machine Learning (ICML'96)*, pages 346 – 354, San Francisco, CA, July 1996. Morgan Kaufmann.

[10] G. A. Stephen. *String Searching Algorithms*. World Scientific Publishing, Singapore, 1994.

[11] J. T.-L. Wang, G.-W. Chirn, T. G. Marr, B. Shapiro, D. Shasha, and K. Zhang. Combinatorial pattern discovery for scientific data: Some preliminary results. In R. Snodgrass and M. Winslett, editors, *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'94)*, pages 115 – 125, Minneapolis, MI, June 1994. ACM.