

Discovery of Serial Episodes from Streams of Events

Taneli Mielikäinen

HIIT Basic Research Unit
Department of Computer Science
University of Helsinki, Finland

1 Introduction

A very important problem in data mining is finding patterns from sequential data. There is a vast number of sources for sequential data such as biological sequences, text documents, telecommunication alarm sequences, click streams, market basket data, web logs, and other time series. One of the most popular patterns mined from sequential data are the episodes, i.e., directed acyclic graphs with labeled nodes [3]. An important subclass of episodes are the serial episodes, which are essentially sequences. Serial episodes are useful in many applications, including network monitoring and molecular biology.

Currently, there are many situations where so much sequential data is produced that it cannot even be stored without great difficulties. That kind of sequential sources are called data streams. In this paper we focus on finding serial episodes from data streams. To the best of our knowledge the problem of mining serial episodes from data streams has been studied in depth only for length-1 episodes [2].

2 Concepts

Definition 1 (Sequence and stream) Let \mathcal{I} be a set of items. A sequence s over \mathcal{I} is an ordered list $s[1]s[2] \dots s[|s|] = s[1, |s|]$ of elements. The length of the sequence s , i.e., number of items in the ordered list s is denoted by $|s|$. A sequence of length k is called a k -sequence. An infinite sequence is called a stream. The empty sequence, i.e., the sequence of length zero is denoted by λ .

Usually there is more precise time information available about the events in the stream than just the order they are in the stream. In the context of this paper, this time information can be ignored. There are many ways how the number of occurrences of a sequence t in a sequence s are counted [1, 3]. The notion of an occurrence of t in s , however, is virtually always based on t being a subsequence of s .

Definition 2 (Subsequence) A sequence t is contained in a sequence s , i.e., t is a subsequence of s , if and only if

$|t| \leq |s|$ and there are $1 \leq i_1 < \dots < i_{|t|} \leq |s|$ such that $s[i_j] = t[j]$ for all $1 \leq j \leq |t|$. The sequence t is a proper subsequence of s if t is a subsequence of s and $t \neq s$. We write $t \sqsubseteq s$ ($t \sqsubset s$) if t is a subsequence (proper subsequence) of s . Furthermore, the subsequence of s without the i th item in s is denoted by $s[-i] = s[1, i-1]s[i+1, |s|]$.

The occurrences of t in s form the *cover* of t in s that consists of the index sets of occurrences of t in s , i.e., $cover(t, s) = \{\{i_1, \dots, i_{|t|}\} : t = s[i_1] \dots s[i_{|t|}]\}$. The cardinality of the cover is called the *support* of t in s and denoted by $supp(t, s) = |cover(t, s)|$.

A vital property for most search procedures to find interesting patterns from data is the anti-monotonicity of the interestingness measure which is support in our case. Unfortunately, the support based on the number of occurrences is not anti-monotone:

Example 1 (loss of anti-monotonicity) Let $s = AAAA$. Then $supp(A, s) = 4$ but $supp(AA, s) = 6$. I.e., $supp(A, s) < supp(AA, s)$ and thus the support is not anti-monotone.

It is not easy to achieve anti-monotone definition for the frequency, especially without detailed information about the sequence. The function $fr(t, s) = supp(t, s) / (|s| - 1) \dots (|s| - |t|)$ is anti-monotone as shown in the proof of Theorem 1:

Theorem 1 If a sequence t is contained in a sequence u then the frequency of t in s is not smaller than the frequency of u in s for any sequence s . That is, if $t \sqsubseteq u$ then $fr(t, s) \geq fr(u, s)$ for all sequences s .

Proof. It is sufficient to consider the case where $|t| = |u| - 1$ and $t \sqsubset u$. Each occurrence I of u in s is an extension of an occurrence I' of t in s such that $I \setminus I' = \{i\}$ for some $i \in [|s|]$. The occurrence I' of t can be extended to at most $|s| - |t|$ occurrences I of u since $I' \subset I$ and $I \setminus I' = \{i\}$ with $i \in [|s|] \setminus I'$. \square

Note that the growth speed of the divisor $(|s| - 1) \dots (|s| - |t|)$ cannot be modified without

losing the anti-monotonicity property since the ratio of $\text{supp}(tA, s)$ and $\text{supp}(t, s)$ can be $|s| - |t|$ for some possible extension tA of t :

Example 2 (tightness of the divisor) Let $t = A_1 \dots A_{|t|}$, $u = tA_{|u|}$ and $s = tA_{|u|}^{|s|-|t|}$. Then $\text{supp}(t, s) = 1$, $\text{supp}(u, s) = |s| - |t|$ and thus $\text{supp}(u, s) / \text{supp}(t, s) = |s| - |t|$.

3 Mining the Stream

Let us assume, without loss of generality, that the s is a finite sequence. Counting the number of occurrences of t in s is equal to counting the number of index sets $I = \{i_1, \dots, i_{|t|}\}$ such that $t = s[i_1] \dots s[i_{|t|}]$. Thus, the support of t in a sequence collection \mathcal{S} is at most $|\mathcal{S}|$ whereas the support for t in a single sequence s can be even $\binom{|s|}{|t|}$ which is between $(|s|/|t|)^{|t|}$ and $(e|s|/|t|)^{|t|}$. Hence, in the worst case, enumerating all occurrences of t in s can be exponential in $|s|$ and $|t|$. Fortunately the counting of the occurrences is not so difficult:

Theorem 2 *The number of subsequences t in a sequence s that end at position $i \in [|s|]$ of s is equal to the number of subsequences $t[1, |t|-1]$ in $s[1, i-1]$ if $t[|t|] = s[i]$ and zero otherwise. (The number of subsequences λ in any sequence considered to be one.)*

Proof. If $t[|t|] \neq s[i]$ then the number of subsequences t ending at position i of s is clearly zero. Thus, let us assume that $t[|t|] = s[i]$. All occurrences of t in s ending at position i of s have index sets of form $\{i_1, \dots, i_{|t|}\}$ with $i_1 < \dots < i_{|t|} = i$. Thus, an occurrence of t in s ending at position i is an occurrence of t in $s[1, i]$. Furthermore, each occurrence of $t[1, |t|-1]$ in $s[1, i-1]$ is a prefix of exactly one occurrence of t . Namely an occurrence $s[i_1] \dots s[i_{|t|-1}] = t[1, |t|-1]$ in $s[1, i-1]$ is the prefix of $s[i_1] \dots s[i_{|t|-1}]s[i] = t$ in s . Thus, the number of occurrences of $t[1, |t|-1]$ in $s[1, i-1]$ is equal to the number of occurrences of t in s ending at position i of s . \square

As the number of occurrences of t in s is the sum of the number of occurrences of t in s ending at positions $1, \dots, |s|$, the supports of subsequences in s can be computed by Algorithm 1. As each element of s is needed only once, Algorithm 1 is suitable for mining also data streams.

Clearly, the proposed version of the algorithm is not yet very practical since the total length of all subsequences of s is always at least $|s|$. Defining a minimum support threshold constraint does not make much sense for data streams. Also the minimum frequency threshold has problems since it implicitly requires the data stream to be a stationary source. However, the number of subsequences in a long sequence is likely to be quite large.

Algorithm 1 Counting subsequences in a sequence s .

```

1: function COUNTSUBSEQUENCES( $s$ )
2:    $\mathcal{P} \leftarrow \{\lambda\}$ 
3:   for  $i = 1, \dots, |s|$  do
4:     for  $r \in \mathcal{P}$  in descending order in lengths do
5:       if  $rs[i] \notin \mathcal{P}$  then
6:          $\mathcal{P} \leftarrow \mathcal{P} \cup \{rs[i]\}$ 
7:       end if
8:        $\text{supp}(rs[i]) \leftarrow \text{supp}(rs[i]) + \text{supp}(r)$ 
9:     end for
10:  end for
11:  return  $\text{supp}$ 
12: end function

```

An alternative to support-based pruning is to restrict the search space by structural constraints. Probably the simplest way is to constraint the maximum length of the sequence of interest. This constraint can be clearly embedded into Algorithm 1.

The method can be adapted also to more complex structural constraints. One very important class of constraints are the maximum gap constraints. A maximum gap constraint g restricts the longest allowed gap between the consecutive indices in the occurrence $\{i_1, \dots, i_{|t|}\}$ of a sequence t in a sequence s , i.e., g restricts the $|i_j - i_{j-1}| = i_j - i_{j-1} \leq g$ for each $j = 2, \dots, |t|$. Let $\text{supp}(t, s, i)$ denote the number of occurrences of t in s that end at the position i of s . The support of a sequence t in s with maximum gap constraint g can be determined by the following recursive formula:

$$\text{supp}(t, s, i) = \begin{cases} \sum_{j=1}^{g+1} \text{supp}(t[-|t|], s, i-j) & \text{if } |t| > 1 \\ \mathbb{1}_{t=s[i]} & \text{otherwise} \end{cases}$$

Thus, the number of counters that have to be maintained in a vertex is maximum length g of the gap plus the depth of the subtree of the vertex. Hence, it is possible to count the supports of the subsequences with maximum gap constraints. The method can be adapted also to sliding windows and the precise time information in the event stream can be taken into consideration, too.

References

- [1] G. Casa-Garriga. Discovering unbounded episodes in sequential data. In N. Lavrac, D. Gamberger, L. Todorovski, and H. Blockeel, editors, *Knowledge Discovery in Databases: PKDD 2003*, volume 2838 of *Lecture Notes in Artificial Intelligence*, pages 83–94. Springer-Verlag, 2003.
- [2] R. M. Karp, S. Shenker, and C. H. Papadimitriou. A simple algorithm for finding frequent elements in streams and bags. *ACM Transactions on Database Systems*, 28(1):51–55, 2003.
- [3] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.