Noname manuscript No.
(will be inserted by the editor)

Ranking Episodes using a Partition Model

Nikolaj Tatti

the date of receipt and acceptance should be inserted later

Abstract One of the biggest setbacks in traditional frequent pattern mining is that overwhelmingly many of the discovered patterns are redundant. A prototypical example of such redundancy is a freerider pattern where the pattern contains a true pattern and some additional noise events. A technique for filtering freerider patterns that has proved to be efficient in ranking itemsets is to use a partition model where a pattern is divided into two subpatterns and the observed support is compared to the expected support under the assumption that these two subpatterns occur independently.

In this paper we develop a partition model for episodes, patterns discovered from sequential data. An episode is essentially a set of events, with possible restrictions on the order of events. Unlike with itemset mining, computing the expected support of an episode requires surprisingly sophisticated methods. In order to construct the model, we partition the episode into two subepisodes. We then model how likely the events in each subepisode occur close to each other. If this probability is high—which is often the case if the subepisode has a high support—then we can expect that when one event from a subepisode occurs, then the remaining events occur also close by. This approach increases the expected support of the episode, and if this increase explains the observed support, then we can deem the episode uninteresting. We demonstrate in our experiments that using the partition model can effectively and efficiently reduce the redundancy in episodes.

1 Introduction

Pattern mining is one of the most well-studied subfields in exploratory data analysis. One of the major setbacks of traditional frequent pattern mining

N. Tatti
HIIT, Department of Information and Computer Science, Aalto University, Finland
E-mail: nikolaj.tatti@aalto.fi

techniques is that the obtained results are heavily redundant. Hence, the focus of the pattern mining field has moved away from mining patterns efficiently to reducing redundancy of the output. This has been especially the case for mining itemsets.

A technique to reduce redundancy that has proved to be efficient for itemsets is to use a partition model [17]. A partition model for itemsets involves in dividing an itemset, say Z, into two subitemsets, say X and Y, and assume that items in X and Y are independent. If the observed support of Z is close to the expected support, then we deem Z uninteresting. In order to select X and Y we simply iterate over all possible partitions and pick the one that fits the best with the observed data. For example, if Z is an itemset that contains an interesting pattern *and* some independent noise events, then the partition model is able to detect this and downplay the importance of Z.

In this paper our goal is to reduce redundancy in episodes, a very general class of sequential patterns [9]. Essentially, an episode is a set of events that should occur in a sequence. In addition, these events may have constraints on the order in which they should occur. This order is expressed by a directed acyclic graph (DAG).

While ranking and filtering patterns to reduce redundancy is well-studied for itemsets, it is surprisingly underdeveloped for episodes. The most straightforward approach to rank episodes is to compare them against the independence model [3, 7, 11]. In this paper we will introduce ranking technique based on partition models instead of independence model. Our goal is that by using partition models we will be able to reduce redundancy in episodes in a similar fashion that partition models allow us to reduce redundancy in itemsets [17].

Computing the expected support for an episode is a more intricate process than computing the expected support for an itemset. For example, to obtain the expected support of an itemset according to the independence model we can simply multiply the margins of individual items. On the other hand, to compute the expectation for an episode, we need to construct a finite state machine, where each state represents the episode events that we have seen so far (see Section 4 for more details). We can then compute the expected support by computing the probability of a random sequence reaching the final state of the machine.

We will consider two types of partition models. In the first approach we partition the episode into two subepisodes. If one or both of these subepisodes have few gaps, then we will increase the probability of a whole subepisode to occur in a sequence once we have seen at least one event from the subepisode. This will increase the expected support of the episode. In the second approach we try to explain the support of an episode with an episode that has the same events but impose more strict constraints on the order. In this case we will increase the probability of events whenever they obey the more strict order.

Fortunately, we can construct the partition model for both aforementioned cases using the same finite state machine that we use to compute the expectation for the independence model. Roughly speaking, when computing the probability of reaching the final state of the finite state machine, we will in-

crease the probability of a random sequence taking certain transitions. These transitions will be determined either by the subepisodes (the first case) or by the superepisode (second case). In both cases, this will increase the probability of a random sequence containing the episode and will increase the expected support of an episode.

The rest paper of the paper is organized as follows. We introduce preliminary notation in Section 2. In Section 3 we describe how to rank episodes given the model. In Section 4 we construct a finite state machine that we need to compute the independence model. Our main methodological contribution is given in the next two sections. In Section 5 we obtain a partition model by boosting certain transitions of the finite state machine. We introduce the partition model using subepisodes and superepisodes in Section 6. We discuss the related work in Section 7. Finally, we introduce our experimental evaluation in Section 8 and conclude the paper with discussion in Section 9.

2 Preliminaries

We begin by introducing the notation that we will use throughout the paper.

Our input dataset consists of m sequences $\mathcal{S} = S_1, \ldots, S_m$. Each sequence contains events coming from some finite universe, which we will denote by Σ.

We are interested in episodes introduced by Mannila et al. [9] and defined as follows.

Definition 1 An *episode* $G = (V, E, lab)$ is a directed acyclic graph with labelled vertices. The labels are represented by the label function $lab : V(G) \to \Sigma$, mapping each vertex to a label.

We will call G a *parallel* episode if G has no edges. On the other hand, an episode that represents a total order is called a *serial* episode.

Informally, an episode represents a set of events that should occur in the order that is consistent with the edges. More formally:

Definition 2 Given a sequence $S = s_1, \ldots, s_n$ and an episode $G = (V, E, lab)$, we say that S *covers* G if there is an injective mapping m from the vertices of G to the indices of s, $m : V(G) \to 1, \ldots, n$, such that

1. labels are honored, $s_{m(v)} = lab(v)$ for every $v \in V$,
2. edges are honored, $m(v) < m(w)$ if $(v, w) \in E$.

Example 1 Consider an episode G_1 given in Figure 1. Definition 2 implies that a sequence S covers G_1 if and only if S contains a followed by b and c in arbitrary order, and finally followed by d, with any number of events before between, or after these 4 events. For example, $aebfcd$ covers G_1 due to a subsequence $abcd$ but $cabde$ does not since there is no c between a and d.

Note that an episode and its transitive closure represent essentially the same pattern: an episode is covered if and only its transitive closure is covered.

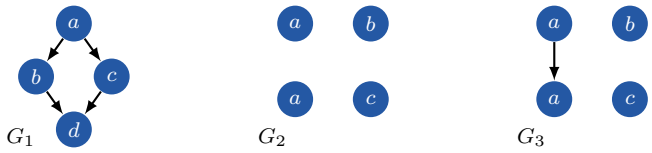

Fig. 1 Toy episodes used in examples

For simplicity, we will assume that we are only dealing with transitively closed episodes. However, for aesthetic reasons, whenever showing an episode we will remove edges that are implied by the transitive closure.

Now that we have defined occurrence in a single sequence, we can define the support of an episode.

Definition 3 Let G be an episode and $\mathcal{S} = S_1, \ldots, S_m$ be a collection of sequences. The *support* of G is the number of sequences covering G,

$$sup(G) = |\{i \mid S_i \text{ covers } G\}| \quad .$$

Since the support is monotonically decreasing, discovering *all* episodes whose support is higher than some given threshold can be done efficiently using APRIORI or DFS style approach.

Unlike itemsets, episodes are surprisingly difficult to handle. For example, checking whether a sequence covers an episode is in fact an **NP**-hard problem [12].

We will focus on a more simple class of episodes, which are called strict episodes [13].

Definition 4 An episode $G = (V, E, lab)$ is *strict* if any two distinct vertices $v, w \in V$ with the same label, $lab(v) = lab(w)$ we have either $(v, w) \in E$ or $(w, v) \in E$.

The need for using strict episodes stems from technical details that we will see in later sections. Nevertheless, this class of episodes is large: it contains all serial episodes, all episodes with unique labels. In addition, for every parallel episode G, there is a strict episode H such that a sequence will cover G if and only if the same sequence covers H. To obtain H from G simply connect all vertices with the same label. For example, G_2 in Figure 1 is a parallel episode while G_3 is a strict episode, and a sequence S covers G_2 if and only if S covers G_3 as well.

From now on we will assume that episodes are strict.

We will need a concept of an induced episode which is essentially a standard notion of an induced graph.

Definition 5 Given an episode $G = (V, E, lab)$ and a subset of vertices $W \subseteq V$, we define an induced episode $G(W)$ to be the episode with vertices W and edges

$$E(W) = \{(v, w) \in E \mid v, w \in W\}) \quad .$$

The vertices have the same labels as the vertices in G.

3 Ranking episodes based on expectation

In this section we describe how to rank episodes based on the expected support. We will give the details for computing the expectation in the latter sections.

Formally, consider that we are given an episode G and a dataset of sequences $\mathcal{S} = S_1, \ldots, S_m$. Unlike with itemsets we need to take into account the length of individual sequences as longer sequences have a higher probability to cover an episode. Assume that we have a generative model M for a sequence, that allows us to compute the probability that G occurs in a sequence of a certain length, that is, we can compute

$$p_k = p(X \text{ covers } G \mid |X| = k, M),$$

where X is a random sequence of length k. We will define different variants of M in the next sections.

Let $\mathcal{X}$ be m random sequences, each random sequence having the same length as the input sequence, $|X_i| = |S_i|$. If we assume that each sequence in $\mathcal{X}$ is generated independently, then the expected support of G according to the model is then

$$\mu = \sum_{S \in \mathcal{S}} p_{|S|} \quad .$$

Moreover, we can easily show that the probability that $sup(G)$ is equal to n is

$$p(sup(G; \mathcal{X}) = n \mid M) = \sum_{\substack{\mathcal{T} \subseteq \mathcal{S} \\ |\mathcal{T}| = n}} \prod_{S \in \mathcal{T}} p_{|S|} \prod_{S \in \mathcal{S} \setminus \mathcal{T}} (1 - p_{|S|}),$$

where the sum goes over all subsets of $\mathcal{S}$ of size n. If all sequences are of equal length, then this distribution is in fact a binomial distribution.

Assume that we observe the support to be $sup(G; \mathcal{S}) = n$. Ideally, we would like to compute the rank to be the probability $p(sup(G; \mathcal{X}) \geq n \mid M)$. This value is close to 1 whenever support is low and 0 whenever the support is large. Note that this quantity can be interpreted as a p-value. However, in this work we will not make this interpretation and treat this quantity simply as a rank (see Section 9 for discussion about interpreting this quantity as a p-value). Since in practice most of the values will be very close to 0 we consider the logarithm of the score, that is, we define

$$\begin{aligned} r(G \mid M) &= -\log p(sup(G; \mathcal{X}) \geq n \mid M) \\ &= -\log 1 - \sum_{k=1}^{n-1} p(sup(G; \mathcal{X}) = k \mid M) \quad . \end{aligned}$$

Episodes that have abnormally high support will have a high rank. Computing the rank can be done in $O(n^2 m)$ with a simple recursive equation. However, this may be slow if n, the observed support, is large. Hence, in practice we will use well-known asymptotic estimates for $p(sup(G; \mathcal{X}) \geq n \mid M)$. If n is large

enough, we can estimate the probability with a normal distribution $N(\mu, \sigma)$, where the variance σ^2 is

$$\sigma^2 = \sum_{S \in \mathcal{S}} p_{|S|}(1 - p_{|S|}) \quad .$$

In practice, the input dataset is large enough so that the approximation is accurate *if* μ is not close to 0. If μ is small, say $\mu \leq 10$, this approximation becomes inaccurate. In such cases, a common approach is to estimate the probability with Poisson distribution with a mean of μ.

4 Independence model for episodes

In this section we review how to compute the expected support of an episode using the independence model. The idea of computing the expected support using the independence model was originally done by Gwadera et al. [3]. This approach requires us to construct a certain finite state machine. In later sections we will use this machine to build the partition model.

4.1 Finding episodes with finite state machine

Our first goal is to construct a finite state machine from an episode. This machine has two purposes. Firstly, we can use it to compute the support of an episode. Secondly, we can use it to compute the expected support, either using the independence model, which we will review in Section 4.2 or the partition model which we will introduce in Section 5.

We start with a definition of a prefix graph which will turn out to be the states of our machine.

Definition 6 Given an episode $G = (V, E, lab)$ and a subset of vertices $W \subseteq V$, we say that an induced graph $H = G(W)$ is a *prefix subgraph* if all ancestors of vertices in W are also included in W, that is,

$$v \in W \text{ and } (w, v) \in E \quad \text{implies} \quad w \in W \quad .$$

We will denote the collection of all prefix graphs by $pre(G)$.

Example 2 Consider an episode G given in Figure 2. This episode has 6 prefix graphs $H_1, \ldots, H_6$, given also in Figure 2. Note that the empty graph $G(\emptyset)$ and the full graph G are both prefix graphs.

Now that we have defined the states of our machine, we can finally define the machine itself.

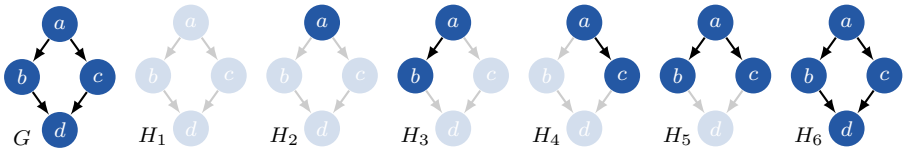

Fig. 2 Toy episode G and all the prefix graphs $H_1, \ldots, H_6$.

Definition 7 Given an episode G we define a *machine* $M(G)$ to be a DAG with labelled edges, such that the states are the prefix subgraphs $pre(G)$ and two states H_1 and H_2 are connected with an edge (H_1, H_2) if we can obtain H_1 by deleting a (sink) vertex from H_2. The label of the edge is the label of the deleted vertex.

The *source state* of $M(G)$ is the empty prefix graph $G(\emptyset)$, while the *sink state* is the episode G itself.

Example 3 Consider an episode G given in Figure 2. This episode has 6 prefix graphs given also in Figure 2 and the machine $M(G)$ is given in Figure 3.

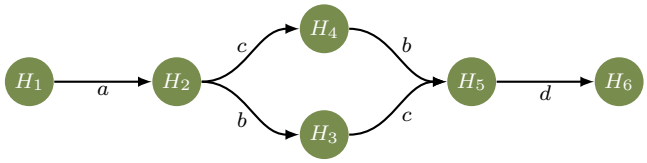

Fig. 3 A machine $M(G)$ for the episode given in Figure 2.

Note that we can view $M(G)$ as a finite state machine with a small technical difference. Finite state machine requires that we should specify transitions from each state for *every possible* label. We can think of $M(G)$ as a finite state machine by adding self-loops for every possible missing label. However, it is more natural to ignore these self-loops from the notation since in practice $M(G)$ is implemented as a DAG.

Our next technical lemma is the key result why we are working only with strict episodes. We will see later on how this lemma helps us with the definitions and propositions.

Lemma 1 *Let G be a strict episode and let H be a state in $M(G)$. Each outgoing edge from H has a unique label among outgoing edges. Each incoming edge to H has a unique label among incoming edges.*

Proof Assume that there are two edges (H, F_1) and (H, F_2) having the same label. This means that there are two distinct vertices v and w in G with the same label such that $H = F_1 \setminus v$ and $H = F_2 \setminus w$. Since G is strict, v and w must be connected. Assume that $(v, w) \in E(G)$. This means that F_2 cannot be a

prefix graph since v is a parent of w and is not in $V(F_2)$. This is a contradiction and shows that every outgoing edge has a unique label. The proof for incoming edges is similar. □

Our next definition is a greedy function mapping a sequence and an initial state to a final state. The final state is essentially a state that we will end up by walking greedily the edges $M(G)$.

Definition 8 Given a machine $M = M(G)$ for a strict episode G, a state H in M, and a sequence $S = s_1, \ldots, s_n$, we define $gr(M, S, H)$ to be the state to which s leads M from H, that is, we can define $gr(M, S, H)$ recursively by first defining gr for the empty sequence, $gr(M, \emptyset, H) = H$, and then more generally, for $i = 1, \ldots, n$,

$$gr(M, s_i, \ldots, s_n, H) = gr(M, s_{i+1}, \ldots, s_n, F)$$

if $(H, F) \in E(M)$ with a label s_i, and

$$gr(M, s_i, \ldots, s_n, H) = gr(M, s_{i+1}, \ldots, s_n, H)\,,$$

otherwise.

We will abbreviate $gr(M, S, G(\emptyset))$ by $gr(M, S)$.

Note that this definition is only well defined if $M(G)$ has unique outgoing edges. Lemma 1 guarantees this since G is a strict episode.

Example 4 Consider $M = M(G)$ given in Figure 3. Then, for example,

$$gr(M, adc) = gr(M, adc, H_1) = H_4 \quad \text{and} \quad gr(M, bcd, H_2) = H_6 \quad .$$

One of the reasons we defined $M(G)$ is the fact that we can use this to detect when a sequence is covering G. First let us define the coverage for a *state* in $M(G)$.

Definition 9 We say that a sequence S covers a state H in $M(G)$ if there is a subsequence T of S leading from the source state to H, that is,

$$gr(M, T) = H \quad .$$

As expected, covering an episode G and the sink state in $M(G)$ are closely related.

Proposition 1 (Proposition 1 in [11]) *Sequence S covers an episode G if and only if S covers the sink state in $M(G)$.*

The technical difficulty with using the definition of coverage is that we need to *find a subsequence* that travels from the source state to the sink state. Fortunately, the next result states that we can simply use the whole sequence.

Proposition 2 (Corollary 1 in [11]) *Sequence S covers the sink state in $M(G)$ if and only if $gr(M, S) = G$.*

For the sake of completeness we provide the proof in the appendix.

Example 5 Consider G given in Figure 2 and its corresponding machine $M = M(G)$ given in Figure 3. Sequence $S = aebfcd$ covers G. Proposition 1 implies that there is a subsequence of S, say T, such that $gr(M, T) = H_6$ and Proposition 2 makes a stronger claim that one can choose $T = S$. By applying the definition of the greedy function, we can easily verify that indeed $gr(M, S) = H_6$.

We should point out that this does not hold for a general finite state machine, however, this holds for any $M(G)$.

4.2 Independence model

Our next step is to compute the expected support. Here we use the results from the previous section, by computing the probability that a random sequence reaches the sink state.

We will use the following notation.

Definition 10 Let $M = M(G)$ be a machine and let H be a state. Let $S = s_1, \ldots, s_n$ be a random sequence of n events, generated independently. Define

$$p_{ind}(H, n) = p(gr(M, S) = H)$$

to be the probability that S leads to H from the source state.

In other words, the probability that a sequence of n events covers G is equal to $p_{ind}(G, n)$.

We can now compute the probability recursively using the following proposition.

Proposition 3 *Let $M = M(G)$ be a machine and let H be a state. Let $S = s_1, \ldots, s_n$ be a random sequence of n events, generated independently. Then the probability of $gr(M, S) = H$ is equal to*

$$p_{ind}(H, n) = q \times p_{ind}(H, n-1) + \sum_{e=(F,H)\in E(M)} p(lab(e)) p_{ind}(F, n-1)\,,$$

where q is the probability of being stuck in H for a single event

$$q = 1 - \sum_{e=(H,F)\in E(M)} p(lab(e)) \quad .$$

This proposition is a special case of Proposition 4, hence we will omit the proof.

If we write M_{ind} to be the independence model, we define

$$r_{ind}(G) = r(G \mid M_{ind}) \quad .$$

To compute this rank we need to compute the probability that a random sequence of length k covers episode G. This is exactly what Proposition 3 does.

Example 6 Assume that the alphabet consists of 5 labels and the probabilities for labels are $p(a) = 0.4$, $p(b) = 0.3$, $p(c) = 0.2$, $p(d) = 0.06$, and $p(e) = 0.04$. Consider M given in Figure 3. The initial probabilities are

$$p_{ind}(H_1, 0) = 1, \quad p_{ind}(H_j, 0) = 0, \quad \text{for} \quad j = 2, \ldots, 6 \quad .$$

According to Proposition 3 the probabilities are

$$\begin{aligned}
p_{ind}(H_1, n+1) &= 0.6 p_{ind}(H_1, n)\,, \\
p_{ind}(H_2, n+1) &= 0.5 p_{ind}(H_2, n) + 0.4 p_{ind}(H_1, n)\,, \\
p_{ind}(H_3, n+1) &= 0.8 p_{ind}(H_3, n) + 0.3 p_{ind}(H_2, n)\,, \\
p_{ind}(H_4, n+1) &= 0.7 p_{ind}(H_4, n) + 0.2 p_{ind}(H_2, n)\,, \\
p_{ind}(H_5, n+1) &= 0.94 p_{ind}(H_5, n) + 0.2 p_{ind}(H_3, n) + 0.3 p_{ind}(H_4, n)\,, \\
p_{ind}(H_6, n+1) &= 0.06 p_{ind}(H_5, n) + p_{ind}(H_6, n) \quad .
\end{aligned}$$

5 Partition model for episodes

Consider an episode G given in Figure 4 and its machine $M(G)$. Assume that b has tendency to occur soon after a but c is a freerider: its occurrence is independent of vicinity of a and b. This episode will have a high rank because its support is higher than what independence model predicts. The reason for this is that b occurs more often than expected after a, that is, we will move sooner from state H_2 to H_3 and from H_5 to H_6 sooner than expected.

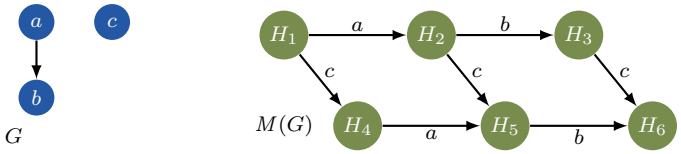

Fig. 4 Toy episode and its machine

Our goal is to construct a more flexible model that would take into account that some of the transitions are more probable than what the independence model predicts. This will allow us to remove the freeriders.

In order to do that let us fix an episode G and assume that we are given two disjoint subsets of edges $C_1 \subset E(M(G))$ and $C_2 \subset E(M(G))$. Note that (both of) these sets can be empty and it is possible that $C_1 \cup C_2 \neq E(M(G))$. We will describe later on how we select these sets but for now we will assume that they are given. Also, we can easily define this model for k sets but we only need two sets.

Our model has $|\Sigma| + 2$ parameters: $|\Sigma|$ parameters u_l states the likelihood of a label l. The larger u_l, the more likely l is to occur in a sequence. In addition, we have two transition parameters. Parameter t_1 states how likely we use an edge in C_1 while t_2 states how likely we use an edge in C_2.

In order to define the actual model let us first define the conditional probability of a label *given* a state H,

$$p(l \mid H) = \begin{cases} \frac{1}{Z_H} \exp(u_l + t_1), & \text{if there is } (H,F) \in C_1 \text{ and } lab((H,F)) = l, \\ \frac{1}{Z_H} \exp(u_l + t_2), & \text{if there is } (H,F) \in C_2 \text{ and } lab((H,F)) = l, \\ \frac{1}{Z_H} \exp(u_l), & \text{otherwise,} \end{cases}$$

where Z_H is a normalization constant guaranteeing that $\sum_l p(l \mid H) = 1$. Note that Z_H depends on H while u_l and t_1 and t_2 do not.

This probability implies that the labels with large u_l are more likely to occur. Moreover, if there is an edge $(H,F) \in C_1$, then the probability of generating the label of the edge is increased due to t_1 (and similarly for C_2).

Note that this is well defined because Lemma 1 states that labels for outgoing edges are unique.

We select u_l and t_i by optimizing the likelihood of a sequence. In order to do this, we first new to define the probability of a sequence. Let us first decompose it into conditional probabilities,

$$p(S) = \prod_{i=1}^{n} p(s_i \mid s_1, \ldots, s_{i-1}) \quad .$$

We define the probability of s_i to be

$$p(s_i \mid s_1, \ldots, s_{i-1}) = p(s_i \mid H),$$

where H is the state given by the greedy function,

$$H = gr(M, (s_1, \ldots, s_{i-1})) \quad .$$

In other words, s_i is generated from $p(\cdot \mid H)$, where H is the current state led by $s_1, \ldots, s_{i-1}$.

Note that if $C_1 = C_2 = \emptyset$, then $p(l \mid H) = p(l)$, and the model is in fact the independence model. However, if C_1 and C_2 are not empty, certain labels are expected to occur more often[1] depending on the current state of $M(G)$.

Our next step is to compute the probability of a sequence covering an episode. To that end, let us define $p_{prt}(H,n)$ to the probability according to the partition model that a random sequence of length n reaches H. The following proposition, a generalization of Proposition 3, allows to compute the expected support.

Proposition 4 *Let $M = M(G)$ be a machine and let H be a state. Let $S = s_1, \ldots, s_n$ be a random sequence of n events, generated independently. Then the probability of $gr(M,S) = H$ is equal to*

$$p_{prt}(H,n) = q \times p_{prt}(H, n-1) + \sum_{e=(F,H) \in E(M)} p(lab(e) \mid F) p_{prt}(F, n-1),$$

[1] or more rarely if t_i are small.

where q is the probability of being stuck in H for a single event

$$q = 1 - \sum_{e=(H,F)\in E(M)} p(lab(e) \mid H) \quad .$$

The proof of this proposition is given in the appendix.

Our final step is to find the parameters $\{u_i\}$, t_1, and t_2 of the model. Here we select the parameters optimizing the likelihood of $\mathcal{S}$

$$p(\mathcal{S}) = \prod_{i=1}^{m} p(S_i),$$

that is we assume that each sequence in $\mathcal{S}$ is generated independently. Unlike with the independence model we do not have a closed solution. However, we can show that the likelihood is a concave function of u_l, t_1 and t_2.

Proposition 5 $\log p(\mathcal{S})$ *is a concave function of the model parameters* $\{u_i\}$, t_1 *and* t_2.

The proof of this proposition is given in the appendix.

The concavity allows us to use gradient methods to find the local maximum which is guaranteed to be also the global maximum. We used Newton-Raphson method to find the optimal solution. The technical details for computing the descent are given in Appendix D.

Example 7 Consider a serial episode $G = a \to b \to c \to x$. Assume that there are no gap events between a and b, and b and c, and x occurs independently of other events.

In such case, the independence model will overestimate the the sizes of gaps between a and b, and b and c. This leads to underestimating the probability G occurring in a sequence of a given length, which ultimately leads to underestimating the support of G.

On the other hand, let us set $C_1 = \{(a, a \to b), (a \to b, a \to b \to c)\}$ and $C_2 = \emptyset$. Then the maximum likelihood solution will have $t_1 = \infty$. This means that the partition model will never generate a gap event between a, b, and c, which implies an increase in the expected support. In fact, since we x assume that x is independent of a, b, and c, the partition model corresponds exactly the generating model, and consequently the estimate of the support is unbiased.

6 Which partition models to use?

Now that we have defined our model for ranking episodes, our next step is to consider which models to use. That is, how to select C_1 and C_2. Here we consider two approaches. In the first approach we consider a partition model rising from a prefix graph and in the second approach we consider a model rising from a superepisode. Finally, we combine both of these approaches in Section 6.3 by selecting the model providing the best explanation for the support.

6.1 Partition model from prefix graphs

Now that we have defined our model, our next step is to select which transitions in $M(G)$ we should boost, that is, how to select C_1 and C_2.

We consider two approaches. The first approach, described in this section, is to divide the episode into two subepisodes. The second that is based on considering superepisodes will be described in the next section.

Informally, our idea is to consider a prefix graph H of G. Every vertex in H corresponds to possibly several edges in $M(G)$. This will give us the first set of edges C_1. These transitions determine the occurrence of H in a sequence. The other set of edges, C_2, is given by the vertices outside H.

In order to define this formally, let us first define the set of edges in $M(G)$ based on a subset of vertices.

Definition 11 Given an episode $G = (V, E, lab)$ and a subset of vertices W, define a subset of edges $C_p(W \mid G)$ of a machine $M = M(G)$,

$$C_p(W \mid G) = \{(H, F) \in E(M) \mid V(H) \cap W \neq \emptyset, V(F) \setminus V(H) \subseteq W\},$$

that is, $C_p(W \mid G)$ contains the edges (H, F) such that

1. F is obtained from H by adding a vertex from W,
2. H contains at least one vertex from W.[2]

Let $G = (V, E, lab)$ be an episode. Given a prefix graph H with a vertex set W, we define two sets of edges as $C_1 = C_p(W \mid G)$ and $C_2 = C_p(V \setminus W \mid G)$. Since our goal is to explain the support of G using *smaller* episodes, we will require that $W \neq \emptyset$ and that $W \neq V$.

Example 8 Consider an episode G given in Figure 2 along with its prefix graphs, and also the corresponding $M(G)$ given in Figure 3. There are four possible prefix graphs $H_2, \ldots, H_5$. These graphs give a rise to the edge sets,

$$\begin{aligned}
H_2: \quad & C_1 = \emptyset, \\
& C_2 = \{(H_4, H_5), (H_3, H_5), (H_5, H_6)\}, \\
H_3: \quad & C_1 = \{(H_2, H_3), (H_4, H_5)\}, \\
& C_2 = \{(H_5, H_6)\}, \\
H_4: \quad & C_1 = \{(H_2, H_4), (H_3, H_5)\}, \\
& C_2 = \{(H_5, H_6)\}, \\
H_5: \quad & C_1 = \{(H_2, H_3), (H_2, H_4), (H_4, H_5), (H_3, H_5)\}, \\
& C_2 = \emptyset \quad .
\end{aligned}$$

Let us consider H_5, an episode, where a is followed by b and c, in any order. Let us assume b and c occurs almost immediately after a, in other words, the edges in C_1 should be traversed quickly, which leads to a large t_1, and elevated

[2] Consequently, F contains at least two vertices from W.

expected support. On the other hand, (H_5, H_6) is not boosted in anyway, that is, we model d independently of a, b, and c.

Let us now take a closer look on $H_3 = a \rightarrow b$. Assume that H_3 has elevated support and the main reason for this elevated support is that b occurs often after a almost immediately. Consider now the corresponding edges C_1 in $M(G)$, (H_2, H_3) and (H_4, H_5). These edges correspond to seeing b after we have witnessed a (in the latter we have also witnessed c as a gap event). Hence, by our assumption these edges should be traversed quickly, that is, t_1 should be large. Similarly, C_2 corresponds to $c \rightarrow d$, and if d occurs often c, then t_2 should also be large. Consequently, if t_1 and/or t_2 is large, then the model will yield an increased expected support for G.

Note that in the definition of $C_p(W \mid G)$ we require that the parent node of an edge must be a state containing at least one member in W. For example, outgoing edges of the source state of $M(G)$ will never be a part of C_1 or C_2. The idea behind this constraint is that C_1 and C_2 should not model the likelihood of finding the first vertex of the prefix graph (or the postfix graph). Instead we want model how likely we will find the remaining vertices of an episode once the first vertex is found in a sequence.

To justify the definition of $C_p(W \mid G)$, consider a machine $N = M(G(W))$. An abnormally large support of $G(W)$ suggests that the edges in N are traversed abnormally fast, that is, the number of gap events is low. As the following proposition states the edges in $C_p(W \mid G)$ have a direct correspondence to the edges in N, and so they will be traversed abnormally fast. By modelling this phenomenon with a parameter t_1, we hope to take into account the large support of $G(W)$.

Proposition 6 *Let $G = (V, E, lab)$ be an episode. Let $W \subseteq V$ be a subset of vertices such that $G(W)$ or $G(V \setminus W)$ is a prefix subgraph. Let $M = M(G)$ and $N = M(G(W))$. Define a mapping ρ from states of M to states of N to be $\rho(H) = H(V(H) \cap W)$. Then ρ is a surjection and for any edge in $(H, F) \in E(M)$ one of the following holds*

1. *$\rho(H) = \rho(F)$ or $\rho(H)$ is the initial state or*
2. *$(\rho(H), \rho(F))$ is an edge in N and $(H, F) \in C_p(W \mid G)$.*

In addition, for every edge $(H', F') \in N$ there is an edge $(H, F) \in C_p(W \mid G)$ such that $H' = \rho(H)$ and $F' = \rho(F)$.

Proof Assume that $G(W)$ is a prefix subgraph (the $G(V \setminus W)$ case is similar). A state H in N corresponds to a prefix subgraph of $G(W)$, which makes H also a prefix graph of G, and, by definition, a state in M. The fact that $\rho(H) = H$, makes ρ a surjection.

Let H and F be two states in M such that $(H, F) \in E(M)$. Then F is obtained from H by adding one vertex, say w. If $w \notin W$, then $\rho(H) = \rho(F)$. Assume that $w \in W$ and $\rho(H)$ is not the initial state, that is, $H \cap W \neq \emptyset$. Then, by definition, $(\rho(H), \rho(F))$ is an edge in N and $(H, F) \in C_p(W \mid G)$

The last statement follows immediately from the fact that $\rho(H) = H$ whenever H is a prefix subgraph of $G(W)$. $\square$

Let H be a prefix graph of G and let C_1 and C_2 be the edges as constructed above. Write $M(C_1, C_2)$ to be the partition model. We define the rank to be

$$r_{prt}(G; H) = r(G \mid M(C_1, C_2)) \quad .$$

This rank can be computed using Proposition 4. In our experiments, we mimic approach by Webb [17] for itemsets and use the smallest rank among all possible prefix graphs, see Section 6.3 for more details.

We should point out that from technical point of view, H in $r_{prt}(G; H)$ does not need to be a prefix graph. However, models that are generated from non-prefix graphs may behave unexpectedly.

Example 9 Consider an episode $G = a \to b \to c$ and let $W = \{a, c\}$. The machine $M(G)$ consists of 4 states $H_1 \to H_2 \to H_3 \to H_4$, and $C_1 = C_p(W \mid G) = (H_3, H_4)$ and $C_2 = \emptyset$. Note that in this case t_1 does *not* model the number of gaps between a and c, instead it models the number of gaps between b and c. In fact, if we set $W' = \{b, c\}$, then $C_1 = C_p(W' \mid G)$ and $C_2 = C_p(\{a\} \mid G)$.

The essential problem shown in the example is that there is no direct transition in $M(G)$ of observing c after we have seen a. The following proposition shows that this problem can be prevented if and only if we use prefix graphs.

Proposition 7 *Let G be an episode and let $M = M(G)$ be the corresponding machine. Let W_1 be a subset of nodes, and let $W_2 = V \setminus W_1$. Then the following statements are equivalent:*

1. *either W_1 or W_2 induces a prefix graph.*
2. *for any $X \in pre(G)$ and $i = 1, 2$ such that $\emptyset \neq V(X) \cap W_i \neq W_i$ there exists $Y \in pre(G)$, depending on X and i, such that $(X, Y) \in C_p(W_i \mid G)$.*

Proof The direction *(1) → (2)* is trivial. Let us prove the other direction. Assume that neither W_1 nor W_2 induce a prefix graph. Then there is $v \in W_2$ and $w \in W_1$ such that $(v, w) \in E$. Let X be the largest prefix graph not containing v, such graph exists as the union of the two prefix graphs is a prefix graph.

Assume that $V(X) \cap W_1 \neq \emptyset$. Since $w \notin V(X)$, we also have $V(X) \cap W_1 \neq W_1$. Assume that that there is $Y \in pre(G)$, such that $(X, Y) \in C_p(W_1 \mid G)$. By definition, Y is obtained from X by adding a vertex from W_1. Since Y also does not contain v, this violates the maximality of X.

Assume that $V(X) \cap W_1 = \emptyset$. This is possible only if $(v, u) \in E$ for every $u \in W_1$. Since W_2 does not induce a prefix graph, there is $a \in W_1$ and $b \in W_2$ such that $(a, b) \in E$. Define X' to be the maximal prefix graph not containing a. This graph contains v and does not contain b. Consequently, $W_2 \neq V(X') \cap W_2 \neq \emptyset$. Assume that that there is $Y \in pre(G)$, such that $(X', Y) \in C_p(W_2 \mid G)$. By definition, Y is obtained from X by adding a vertex from W_2. Since $a \notin V(Y)$, this violates the maximality of X'. □

6.2 Partition model from superepisodes

In the previous section we considered a model predicting the support of an episode based on two smaller episodes. In this section we approach the ranking from another perspective. Namely, we try to predict the support of G using superepisodes of G.

In order to motivate this consider the following example.

Example 10 Consider two episodes G_1 and G_2 given in Figure 5. Episode G_2 is a superepisode of G_1. Assume that in our dataset, event b occurs often once a has occurred. This is to say that if we are in H_2 in either $M(G_1)$ or $M(G_2)$ we are likely to move soon to H_4.

Assume also that occurrence of a after b follows the independence model, or that b occurs rarely without a in front of it. In both cases the elevated support of G_1 can be explained by the fact that b follows often after a. This means that we can explain the elevated support of G_1 if we know that G_2 has an elevated support, and consequently we should assign G_1 a low rank.

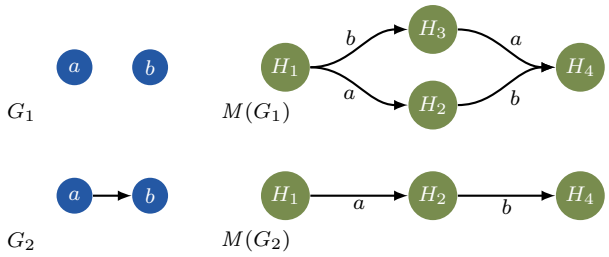

Fig. 5 Episode G_1 and G_2 and the corresponding machines $M(G_1)$, $M(G_2)$.

Assume two episodes $G = (V, E_1, lab)$ and $H = (V, E_2, lab)$ such that $E_1 \subsetneq E_2$. If $(W, E_2(W))$ is a prefix graph of H, then $(W, E_1(W))$ is also a prefix graph of G. This allows us to define a mapping ρ from $pre(H)$ to $pre(G)$ by setting $\rho((W, E_2(W))) = (W, E_1(W))$. Moreover, if x is a sink in $(W, E_2(W))$, then it is also a sink in $(W, E_1(W))$. This immediately implies ρ can be viewed as a graph homomorphism from $M(H)$ to $M(G)$, essentially making $M(H)$ a subgraph of $M(G)$. We can now define the set of edges of $M(G)$ for our partition model to be the edges in $M(H)$. More formally,

Definition 12 Given two episodes $G = (V, E_1, lab)$ and $H = (V, E_2, lab)$ such that $E_1 \subsetneq E_2$, define a subset of edges $C_s(H \mid G)$ of a machine $M = M(G)$,

$$C_s(H \mid G) = \{\rho(X, Y) \in E(M) \mid X \neq \emptyset, (X, Y) \in E(M(H))\},$$

that is, $C_s(H \mid G)$ contains the edges from non-source vertices that can be also found in $M(H)$.

We can now define $C_1 = C_s(H \mid G)$ to be the first set of edges and $C_2 = \emptyset$. Note that, similarly to the prefix graph approach from the previous section, C_1 will not contain any edges from the source state. The rationale here is the same: transitions from the source state indicate beginning of an episode while we are interested in modelling how fast we can find the complete episode once we have found the first label. Also note that since we require that $E_1 \neq E_2$, we will have at least one edge $(H, F) \in E(M(G))$ such that $H \neq \emptyset$ and (H, F) is not contained C_1.

Example 11 Consider an episode G given in Figure 2 and also the corresponding $M(G)$ given in Figure 3.

Assume a serial episode $H = a \to b \to c \to d$. Then

$$C_s(H \mid G) = \{(H_2, H_3),\, (H_3, H_5),\, (H_5, H_6)\}\,,$$

where H_i are given in Figure 3.

On the other hand, if we set $H = a \to c \to b \to d$. Then

$$C_s(H \mid G) = \{(H_2, H_4),\, (H_4, H_5),\, (H_5, H_6)\} \quad .$$

Let H be a superepisode of G and let C_1 be the edges as constructed above. Write $M(C_1, \emptyset)$ to be the partition model. We define the rank to be

$$r_{prt}(G; H) = r(G \mid M(C_1, \emptyset)) \quad .$$

This rank can be computed using Proposition 4. In our experiments, we use the smallest rank induced by a superepisode in our candidate set.

6.3 Combining ranks

Now that we have defined several different partition models, we propose a simple approach to combine these models into a single rank.

To that end, assume that we have a collection $\mathcal{C}$ of episodes that we wish to rank. These candidate episodes are obtained, for example, by mining frequent closed episodes. For a given episode $G \in \mathcal{C}$, let $\mathcal{P} = pre(G) \setminus \{G(\emptyset), G\}$ be the prefix graphs without the empty or the full prefix graph. Also, let $\mathcal{Q}$ be the proper superepisodes of G in $\mathcal{C}$ having the same vertices as G. We then compute the rank by taking the smallest rank among all partition models,

$$r_{prt}(G) = \min(\min_{H \in \mathcal{P}} r_{prt}(G; H)\,, \min_{H \in \mathcal{Q}} r_{prt}(G; H)) \quad .$$

That is, if we can explain the support of G by either a single prefix model or a single superepisode in Q, then we will deem G as redundant.

This approach mimics the approach of Webb [17], where itemsets are filtered by comparing the observed support against the best 2-partition model.

Computational complexity Finally, let us conclude this section with a short discussion about computational complexity. Assume that we have an episode G with n nodes. Let $m = |E(M(G))|$ be the number of edges in $M(G)$.

Using the partition model is a two-step process, the first step is to find the parameters while the second step is to compute the rank. The first step uses iterative gradient descent, for example, Newton-Raphson descent that requires $O(n^{2.373})$ time for Hessian inversion and $O(n^2 + m)$ time for constructing the matrix and gradient. The dominating term will depend on structure of the episode. For example, for serial episodes we have $m = n$. For general episodes we must have $m \leq 2^n$ and for parallel episodes we have $m = 2^n$.

In order to compute $r_{prt}(G)$ we need to loop over all prefix episodes. Again, the number of such episodes depends on G. For serial episodes there are only $n + 1$ such episodes, whereas a parallel episode has 2^n prefix episodes. The parallel episode case is the worst case since there are only 2^n subepisodes in any G.

This implies that in theory computing this rank may not scale for large episodes, especially if they are parallel. Fortunately, in practice, most episodes are small and for these cases our approach remains feasible.

7 Related Work

Discovering episodes: Episode discovery was introduced by Mannila et al. [9] where the authors consider episodes defined as DAGs and consider two concepts of support: the first one based on sliding windows of fixed length and the second one based on minimal windows. Unfortunately, the number of minimal windows is not monotonic in general—however this can be fixed by considering the maximal number of non-overlapping windows, see for example [6]. Mining general episodes can be intricate and computationally heavy, for example, discovering whether a sequence covers a general episode is **NP**-hard [12]. Consequently, research focus has been into mining subclasses of episodes, such as, episodes with unique labels [1, 10], and strict episodes [13]. A miner for general episodes that can handle simultaneous events was proposed by Tatti and Cule [12]. An important subclass of episodes are serial episodes or sequential patterns. A widely used miner for mining closed serial episodes was suggested by Wang and Han [15].

Ranking episodes: Unlike with itemset mining, ranking episodes based on surprisingness is underdeveloped. The most straightforward way of ranking episodes, reviewed in Section 4, by comparing the support against the independence model, was introduced by Gwadera et al. [3]. Using Markov models instead of the independence model to rank serial episodes was suggested by Gwadera et al. [2]. Both of these pioneer works focus on ranking episodes by analyzing support based on a sliding window, that is, the input dataset is a single sequence and the support of an episode is the number of sliding windows of fixed length that cover the episode. Interestingly enough, this scenario generates technical complications since the windows are no longer independent,

unlike in the setup where we have many sequences and we assume that they are generated independently. These complications can be overcome but they require additional computational steps. Instead of using windows of fixed length, ranking based on minimal window lengths with respect to the independence model was suggested by Tatti [11]. Ranking serial episodes allowing multiple labels using the independence model was suggested by Low-Kam et al. [7]. Achar et al. [1] also considered a measure that downranks the episode if there is a non-edge (x, y) that occurs rarely, which suggests that we should augment the episode with the edge (y, x).

In related work, Mannila and Meek [8] consider general episodes as generative models for sequences. They generate short sequences by selecting a subset of events from an episode and ordering events with a random order compatible with the episode. They do not allow gaps and only one pattern is responsible for generating a single sequence.

Finally, SQS and GoKrimp, pattern set mining approaches for discovering serial episodes were respectively introduced by Tatti and Vreeken [14] and by Lam et al. [5]. The idea behind the approach is to find a small set of serial episodes that model the data well. In order to do that the authors constructed a model given a set of episodes and used a posteriori probability of the model to score the episode set. The authors then used a heuristic search to find a set with good episodes. In general, the goal of our approach and the is the same: reducing the redundancy in patterns. From a technical point of view, the approaches are different: in this work we rank episodes based on how surprising their support is while the pattern set mining methods select episodes based on how well we can model the data using the episodes. Moreover, we work with general strict episodes while the current pattern set approaches limit themselves to serial episodes. Extending these miners to general episodes is an interesting future line of research. However, it is highly non-trivial due to the fact that the score, the algorithm for computing the score, and the mining algorithm are specifically designed for serial episodes.

8 Experiments

Datasets: In our experiments we used 3 synthetic datasets and 3 text datasets. The sizes of the datasets are given in Table 1.

The first synthetic dataset, *Plant*, was created as follows. We generated 10 000 sequences of length randomly selected from a uniform distribution between 20 and 30. A single event in each sequence was generated from a uniform distribution of 990 events. We planted two serial episodes and one general in the data. The first episode, a serial episode of 4 vertices was planted with no gaps into a randomly selected sequence 200 times. The second episode, a serial episode of 2 vertices was planted with no gaps into a randomly selected sequence 20 times. The third episode, given in Table 2, was planted 10 times with no gaps, the order of events n and m was picked uniformly. We made

sure that the events used in planted patterns did not occur in the noise. This gave us an alphabet of size 1000.

The second synthetic dataset, *Plant2*, was created as follows. We generated 10 000 sequences of length randomly selected from a uniform distribution between 20 and 30. A single event in each sequence was generated from a uniform distribution of 1000 events. We planted two serial episodes with 3 vertices with no gaps 400 times.

The third synthetic dataset, *Gap*, was created as follows. Similarly to *Plant*, we generated 10 000 sequences of length between 20 and 30. An event in each sequence was generated from a uniform distribution of 996 events. We planted one serial episode of 4 events into the data 200 times. We set the probability of the next event being a noise event to be p, this made the average gap length to be $p/(1-p)$. We varied p from 0 to 0.8 with 0.05 increments. We did not plant events if they did not fit into a sequence.

Our fourth dataset, *Moby*, is the novel Moby Dick by Herman Melville.[3] Our fifth dataset, *JMLR* consists of abstracts of papers from the Journal of Machine Learning Research website,[4] Our final dataset, *Addresses*, consists of inaugural addresses of the presidents of the United States.[5] We processed the datasets by stemming the words and removing the stop words. We further split the text into sequences such that a sequence corresponds to a single sentence.

Dataset	$\lvert\mathcal{S}\rvert$	|events|	σ	$\lvert\mathcal{C}\rvert$	time
Plant	10 000	249 955	10	43 029	56s
Plant2	10 000	249 736	10	46 329	50s
Gap	10 000	250 150	–	–	–
Addresses	5584	62 066	5	19 367	12s
JMLR	5986	75 646	5	49 951	46s
Moby	13 987	105 671	5	17 550	26s

Table 1 Basic characteristics of datasets, frequency thresholds, the numbers of discovered episodes, and running time needed to rank the episodes. The number of events for *Gap* is an average over 17 datasets.

Setup: We mined closed strict episodes from each dataset, except *Gap*, with a miner given by Tatti and Cule [13]. As frequency thresholds we used 5 for text datasets and 10 for the synthetic dataset. The amount of discovered patterns, $\lvert\mathcal{C}\rvert$, is given in Table 1. We then proceeded by ranking each episode first by independence model and then by the partition model.[6]

Results: Our main goal is to compare $r_{prt}(G)$, ranks given by the partition model, against the baseline ranks given by the independence model, $r_{ind}(G)$.

Let us first consider the synthetic dataset *Gap*. We considered ranks for 3 different episodes, given in Figure 6, the planted serial episode G_1, the planted

[3] `http://www.gutenberg.org/etext/15`.

[4] `http://jmlr.csail.mit.edu/`

[5] `http://www.bartleby.com/124/`

[6] The implementation is available at `http://research.ics.aalto.fi/dmg/`.

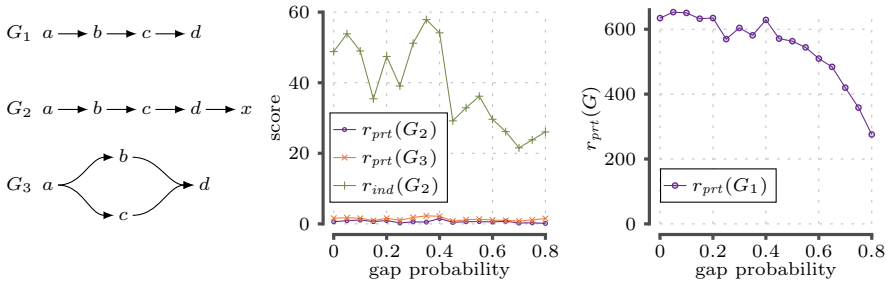

Fig. 6 Episodes and their ranks in *Gap* datasets. The plots for G_2 represent an average of 10 different episodes, each of them having different noise event x. The scales of y-axis of plots are different. The values of $r_{ind}(G_1)$ and $r_{ind}(G_3)$ were outside floating point range.

episode with additional noise event G_2, here we took an average rank of 10 such episodes, and finally G_3 a non-trivial subepisode of G_1. The ranks $r_{ind}(G_1)$ and $r_{ind}(G_3)$ were outside floating point range. The remaining ranks are given in Figure 6 as a function of the gap probability. Let us first consider G_2 and G_3. Unlike the independence model, the partition model predicts the support accurately for these patterns which results in a low rank. Episode G_2 is predicted accurately due to a partition of G_2 to G_1 and the noise label while G_3 is predicted accurately due to G_1 being a superepisode of G_3. As expected, the rank $r_{prt}(G_1)$ remains high as there are no partition model that can explain this pattern. This rank goes down as the average gap length increases as the planted pattern becomes more and more explainable by the independence model.

Let us now look at the top episodes in *Plant* dataset, given in Table 2. The top episode having the largest $r_{ind}(G)$ is the planted serial episode of 4 vertices $a \to b \to c \to d$. The second episode is the planted general episode. However, the next 5 episodes are of form $a \to b \to c \to d \to x$, where x is a noise label. These episodes have abnormally high support because of the original high support of the planted pattern. The 8th episode according to $r_{ind}(G)$ is the second planted episode, namely $e \to f$. Let us now look at the top episodes according to $r_{prt}(G)$. The top 3 episodes are the planted episodes. The remaining episodes are either parallel episodes or serial episodes containing 2 events or episodes of form $a \to b \to c \to d \to x$, where x is a noise label. There is a clear difference between the score values. While the rank for the first three episodes was 78–10^{308}, the ranks for the remaining episodes varied between 0 and 14. In other words, r_{prt} successfully downgraded the freerider episodes that had significant r_{ind}. Some of the freerider episodes still have a significantly large rank. This is due to the multiple hypothesis phenomenon: if we test large amount of patterns, then some of them will have abnormal support just by chance.

We observe similar behaviour in *Plant2* dataset. The top-8 episodes in *Plant2* according to r_{prt} are the 2 planted serial episodes (ranked as 2nd and 4th) and the 6 serial subepisodes with 2 vertices. The ranks of these episodes

Table 2 Top episodes in *Plant* dataset. The symbols x and y represent noise events. The rank for the first episode with respect to the independence model is outside the floating point range.

Independence model			Partition model		
Rank	Episode type	$r_{ind}(G)$	Rank	Episode type	$r_{prt}(G)$
1.	$a \to b \to c \to d$	∞	1.	$a \to b \to c \to d$	10^{308}
2.	$k \to \{n, m\} \to l$	249	2.	$e \to f$	128
3.–7.	$a \to b \to c \to d \to x$	184–185	3.	$k \to \{n, m\} \to l$	78
8.	$e \to f$	128	4.–	$a \to b \to c \to d \to x$	0–14
9.–	$x \to y$ or x, y	2–14		or $x \to y$ or x, y	

Table 3 Kendall-τ coefficients of episodes ranked by r_{prt} and r_{ind}.

Dataset	All	parallel, $V(G) = 2$	$V(G) > 3$
Addresses	0.61	0.60	0.42
JMLR	0.54	0.62	0.45
Moby	0.66	0.59	0.38

were 1205–3704. The remaining episodes were ranked between 0–15. On the other hand, $r_{ind}(G)$ ranked the 2 planted patterns as top-2 episodes. The next 1089 episodes contained either vertices from both patterns, or several vertices from one pattern and one noise event. These episodes had ranks 19–3704. Episodes $G_1 = a \to b \to c \to d \to e \to f$ and $G_2 = (a \to b \to c), (d \to e \to f)$ had ranks $r_{ind}(G_1) = 212$ (17th) and $r_{ind}(G_2) = 289$ (14th) whereas the partition model gave the ranks $r_{prt}(G_1) = 4.8$ (5575th) and $r_{prt}(G_2) = 0.0005$ (45284th). The remaining episodes were ranked between 0–15.

SQS miner [14] discovered the planted serial episode in all *Gap* datasets. In *Plant* SQS discovered the two planted serial episodes, but not the general planted episode, since SQS discovers only serial episodes. Instead, SQS found the two serial superepisodes $k \to n \to m \to l$ and $k \to m \to n \to l$.

Let us now consider episodes discovered from text datasets. In Figure 7 we plot $r_{prt}(G)$ as a function of $r_{ind}(G)$. We highlight parallel episodes with 2 vertices by plotting them separately in the top row while the bottom row contains the episodes with more than two vertices. Note that we omitted serial episodes of size 2 since both ranks will produce an equal score, $r_{ind}(G) = r_{prt}(G)$, since there are no proper superepisodes for an episode G and the only prefix partition is actually equal to the independence model.

The results demostrate that $r_{prt}(G)$ is typically much smaller than $r_{ind}(G)$. This implies that there are lot of patterns whose abnormally high support can be justified by a partition model. In the top row of Figure 7 we see that the parallel episodes of size 2 are typically considered redundant by $r_{prt}(G)$ because typically the serial counterpart of the episode can explain well the behaviour of the parallel episode. For certain parallel episodes, the rank remains the same by design as there are no serial counterpart episodes in the mined collection.

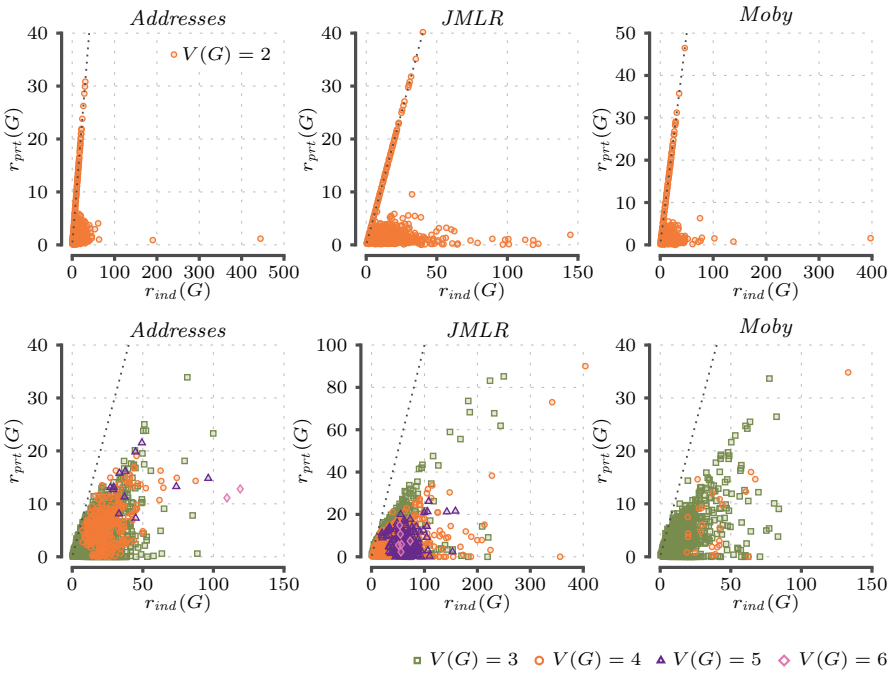

Fig. 7 Partition model ranks $r_{prt}(G)$ as a function of $r_{ind}(G)$ for text datasets. The top row contains parallel episodes with 2 vertices. The bottom row contains episodes with more than two vertices. The ranges of axis vary from figure to figure.

The Kendall-τ coefficients given in Table 3 imply that r_{prt} and r_{ind} are correlated. The correlation is weaker for larger episodes than for parallel episodes of size 2. This is because $r_{prt}(G) = r_{ind}(G)$ if G is a parallel episode of size 2 and does not have a frequent serial episode.

The top episodes according to $r_{prt}(G)$, given in Table 4 in the text datasets were short serial episodes of words that occur often together. This is an expected result as these episodes represent common expressions. For comparison, the top-10 patterns obtained by SQS are given in Table 5. While serial episodes are favored by $r_{prt}(G)$, there are non-serial episodes that have high rank, for example, $G_1 =$ *east*, *west* in *Addresses* has rank $r_{prt}(G_1) = 30$ (42nd), and $G_2 = (subgroup \rightarrow discoveri)$, *rule* in *JMLR* has rank $r_{prt}(G_2) = 25$ (376th).

Our next step is to highlight some episodes that had a high $r_{ind}(G)$ but also ranked low by $r_{prt}(G)$, and vice versa. In order to do that we sorted episodes based on

$$\rho(G) = \frac{r_{ind}(G) - r_{prt}(G)}{r_{prt}(G)} \quad \text{and} \quad \eta(G) = \frac{r_{prt}(G) - r_{ind}(G)}{r_{ind}(G)} \quad . \qquad (1)$$

The top episodes should have large $r_{ind}(G)$ and $r_{prt}(G)$ close to 0. In Figure 8, we listed top-5 episodes from each text dataset. Many of these episodes contain

Table 4 Top-10 episodes according to $r_{prt}(G)$ and $r_{ind}(G)$.

ranked by $r_{ind}(G)$	r_{ind}	r_{prt}	ranked by $r_{prt}(G)$	r_{ind}	r_{prt}
Addresses					
1. unit→state	931	931	unit→state	931	931
2. unit state	445	1.2	fellow→citizen	256	256
3. fellow→citizen	256	256	constitut→state	97	97
4. fellow citizen	190	0.9	four→year	79	79
5. preserv→protect→defend, constitut→unit→state	119	13	men→women	77	77
6. best→abil→preserv→protect, constitut, defend	110	11	year→ago	75	75
7. constitut→unit→state	100	23	armi→navi	63	63
8. constitut→state	97	97	north→south	53	53
9. preserv→constitut→state, protect→defend	96	15	within→limit	52	52
10. unit→state constitut	89	0.6	chief→magistr	51	51
JMLR					
1. support→vector→machin	∞	357	support→vector	440	440
2. support→vector	440	440	support→vector→machin	∞	357
3. support→vector→machin→svm	404	90	support→machin	324	324
4. support→vector→machin svm	356	10^{-3}	vector→machin	306	306
5. reproduc→kernel→hilbert→space	341	73	data→set	284	284
6. support→machin	325	325	real→world	260	260
7. vector→machin	306	306	real→data	213	213
8. data→set	284	284	state→art	191	191
9. real→world	260	260	machin→learn	190	190
10. support→vector→svm	250	85	bayesian→network	166	166
Moby					
1. sperm→whale	874	874	sperm→whale	874	874
2. sperm whale	397	1.6	mobi→dick	359	359
3. mobi→dick	359	359	old→man	224	224
4. old→man	224	224	mast→head	186	186
5. mast→head	187	187	white→whale	179	179
6. white→whale	179	179	right→whale	131	131
7. head mast	138	0.8	quarter→deck	96	96
8. seven→hundr→seventi→seventh	133	35	captain→peleg	86	86
9. right→whale	131	131	chief→mate	85	85
10. old man	102	1.5	new→bedford	82	82

a true pattern, such as, *united* → *states* or *support* → *vector* → *machine* augmented with a common event, seemingly independent event, such as, *world* or *regression*. Let us now compare the top-5 episodes with large $\eta(G)$. Unlike with $\rho(G)$, this list is dominated with episodes for which $sup(G) < \mu_{\text{part}} < \mu_{\text{ind}}$, that is, both methods overestimate the actual support but the partition model is more correct. To make $\eta(G)$ more meaningful, we considered only episodes for which the partition model underestimated the support, $sup(G) \geq \mu_{\text{part}}$, given in Figure 9. We see that the differences between $r_{prt}(G)$ and $r_{ind}(G)$ are small in Figure 9 and large in Figure 8.

Table 5 Top-10 episodes according to SQS. The pattern G in *Moby* was a long episode, such → funni → sporti → gami → jesti → joki → hoki → poki → lad → ocean, a litany repeated 3 times in the novel.

Addresses	*JMLR*	*Moby*
fellow → citizen	support → vector → machin	sperm → whale
unit → state	machin → learn	mobi → dick
men → women	state → art	mast → head
feder → govern	data → set	white → whale
self → govern	bayesian → network	old → man
four → year	larg → scale	captain → ahab
year → ago	nearest → neighbor	G
american → peopl	decis → tree	quarter → deck
vice → presid	cross → valid	right → whale
chief → magistr	neural → network	captain → peleg

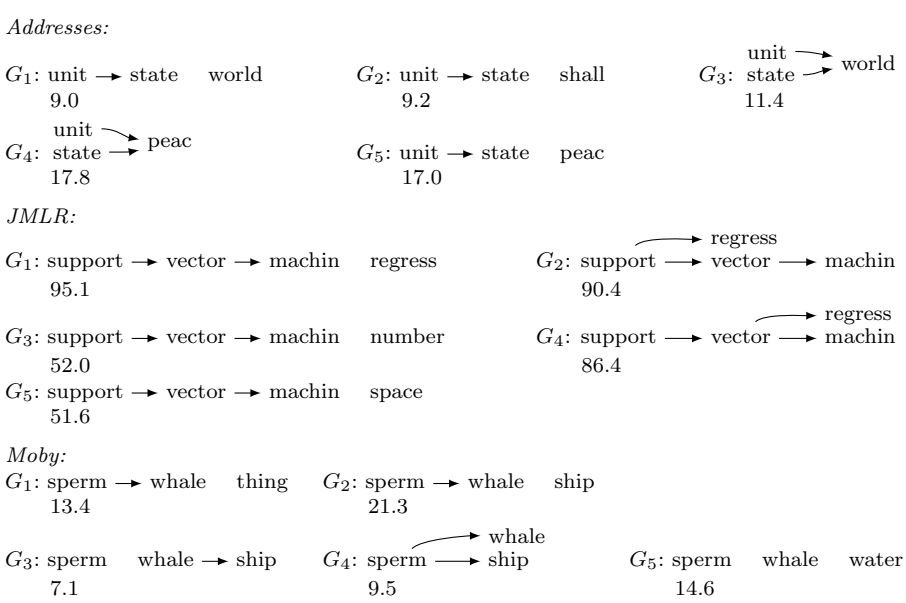

Fig. 8 Episodes with high rank $r_{ind}(G)$ but considered redundant by $r_{prt}(G)$. Top-5 episodes based on $\rho(G)$ given in Eq. 1. The given numbers are $r_{ind}(G_i)$, whereas the partition model ranks are $r_{prt}(G_i) \leq 10^{-6}$.

Many of downgraded episodes are *parallel* episodes, for example, (*united*, *states*), see Table 4. While the independence model ranks them high, the elevated support of a *serial* episode *united* → *states* explains well the elevated support of this parallel episode since these words occur almost always in this particular order. This makes the partition model based on the superepisodes to give this episode a low score.

Finally, let us consider running times that are given in Table 1. We see that we can rank large amount of episodes in a short period of time. Ranking 50 000 episodes took us less than a minute. To obtain a more detailed picture,

Addresses:
G_1: govern great world 1.33 / 1.09 G_2: govern great peac 3.44 / 3.00 G_3: countri nation govern 1.16 / 1.02
G_4: such nation peopl 1.46 / 1.35 G_5: nation govern made 1.62 / 1.52

JMLR:
G_1: algorithm show featur 1.07 / 0.36 G_2: result → model algorithm 1.08 / 0.53 G_3: show → problem, show → algorithm 1.19 / 0.77
G_4: algorithm data obtain 1.08 / 0.75 G_5: model train result 1.47 / 1.21

Moby:
G_1 : round old whale 4.44 / 3.72 G_2 : now though whale 1.68 / 1.62 G_3 : now round whale 1.84 / 1.79
G_4 : out over whale 1.49 / 1.45 G_5 : now whale good 2.26 / 2.21

Fig. 9 Top-5 episodes based on $\eta(G)$ given in Eq. 1, for which partition model underestimated the support. The number format is x/y, where $x = r_{prt}(G_i)$ and $y = r_{ind}(G_i)$.

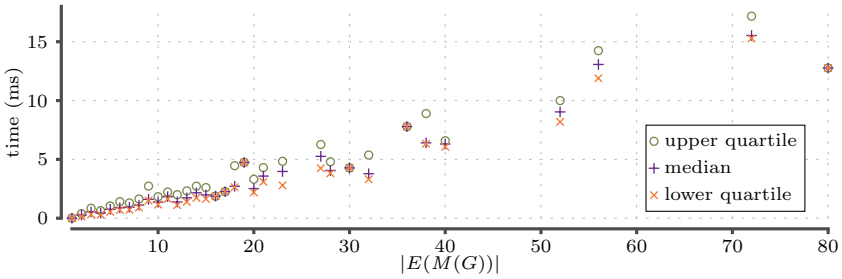

Fig. 10 Time needed to compute the rank for patterns obtained from *JMLR* as a function of the number of edges in $M(G)$.

we present running times as a function of $|E(M(G))|$ in Figure 10 for *JMLR* episodes. We see that the more complex episode (the largest episode contained 5 nodes), the longer it takes to rank. This suggests that while there are complicated steps in computing the support that may even result in exponentially large structures, in practice ranking can be done efficiently.

9 Concluding remarks

In this paper we introduced ranking episodes based on a partition model. Such a ranking reduces redundancy among episodes by ranking episodes low if they can be explained by either two subepisodes or by a more strict episode.

To construct the model we first constructed a finite state machine that is used for computing the expected support for the independence model. We then modified the probabilities of some of the transitions. These transitions are selected based on which subepisodes we are considering. We compare this

model to the independence model and show that for our experiments the model reduces redundancy in patterns.

The effectiveness of the partition model relies on the assumption that the two subepisodes (or the superepisode) have few gaps. This causes the parameters t_1 and t_2 to be large. If this assumption does not hold, that is, $t_1 \approx t_2 \approx 0$, then the partition model will reduce to the independence model. While this assumption is natural and reasonable, in a setup where episodes are frequent but have large gaps, this approach will not reduce redundancy. In such a setup, a different approach is needed, a potential direction for a future line of work.

When partitioning an episode into two subepisodes, we did not consider all the possible partitions. Instead, we only considered partitions arising from prefix graphs. While these partitions are a natural subclass of all possible partitions, this restriction leads to some limitations. For example, we do not partition a serial episode $a \to b \to c \to d$ to $a \to c$ and $b \to d$. However, note that for many episodes, every partition is a partition arising from a prefix graph. This is the case with any parallel episode. We should point out that from technical point of view, we can use non-prefix partitions. However, as demonstrated in Example 9 and Proposition 7, a partition model may not take properly into account the lack of gaps in a non-prefix subepisode. Developing a technique that properly takes interleaving subepisodes into account is an interesting direction for a future work.

Instead of using just the partition model to rank episodes, it may be advantageous to combine it with other ranking method. For example, one approach would be to rank the episodes using the partition model, select top-k episodes, and rerank them based on the independence model. The number k can be given explicitly or determined by interpreting the rank as a p-value, and filtering the episodes based on a given significance level. In the latter approach some extra steps are needed, such as adjusting for the multiple hypotheses testing. This can be done either with direct adjustment or a holdout approach as described by Webb [16]. Strictly speaking, interpreting rank as a p-value requires that we know the exact model parameters which is uncommon. Consequently, in practice and in this work we estimate these parameters, and by doing so estimate the true p-value, by finding the maximum likelihood estimates.

This work opens several future lines of research. One straightforward extension is to combine the partition approach with a markov model suggested by Gwadera et al. [2]. A more intriguing extension is to apply this model for a scenario where we are given one long sequence instead of a database of sequences. In such a case, the support is either based on sliding windows of fixed length or minimal windows. Since the instances are no longer independent, that is, the support is no longer a sum of independent variables, it is likely that we cannot apply the model directly. However, it may be possible to rank episodes by using some other statistic than a support. Table 4 for *JMLR* shows that we can still reduce redundancy among the top patterns. One fruitful approach would be developing a pattern set miner for general episodes. A potential starting point for such a miner could be SQS miner [14], a pattern set miner for serial episodes.

References

1. A. Achar, S. Laxman, R. Viswanathan, and P. S. Sastry. Discovering injective episodes with general partial orders. *Data Mining and Knowledge Discovery*, 25(1):67–108, 2012.
2. R. Gwadera, M. J. Atallah, and W. Szpankowski. Markov models for identification of significant episodes. In *Proceedings of the 5th SIAM International Conference on Data Mining (SDM), Newport Beach, CA*, pages 404–414, 2005.
3. R. Gwadera, M. J. Atallah, and W. Szpankowski. Reliable detection of episodes in event sequences. *Knowledge and Information Systems*, 7(4): 415–437, 2005.
4. S. Kullback. *Information Theory and Statistics*. Wiley, 1959.
5. H. T. Lam, F. Mörchen, D. Fradkin, and T. Calders. Mining compressing sequential patterns. *Statistical Analysis and Data Mining*, 7(1):34–52, 2014.
6. S. Laxman, P. S. Sastry, and K. P. Unnikrishnan. A fast algorithm for finding frequent episodes in event streams. In *Proceedings of the 13th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), San Jose, CA*, pages 410–419, 2007.
7. C. Low-Kam, C. Raïssi, M. Kaytoue, and J. Pei. Mining statistically significant sequential patterns. In *Proceedings of the 13th IEEE International Conference on Data Mining (ICDM), Dallas, TX*, pages 488–497, 2013.
8. H. Mannila and C. Meek. Global partial orders from sequential data. In *Proceedings of the 6th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Boston, MA*, pages 161–168, 2000.
9. 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.
10. J. Pei, H. Wang, J. Liu, K. Wang, J. Wang, and P. S. Yu. Discovering frequent closed partial orders from strings. *IEEE Transactions on Knowledge and Data Engineering*, 18(11):1467–1481, 2006.
11. N. Tatti. Discovering episodes with compact minimal windows. *Data Mining and Knowledge Discovery*, 28(4):1046–1077, 2014.
12. N. Tatti and B. Cule. Mining closed episodes with simultaneous events. In *Proceedings of the 17th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), San Diego, CA*, pages 1172–1180, 2011.
13. N. Tatti and B. Cule. Mining closed strict episodes. *Data Mining and Knowledge Discovery*, 25(1):34–66, 2012.
14. N. Tatti and J. Vreeken. The long and the short of it: summarising event sequences with serial episodes. In *Proceedings of the 18th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Beijing, China*, pages 462–470, 2012.
15. J. Wang and J. Han. Bide: Efficient mining of frequent closed sequences. In *Proceedings of the 20th International Conference on Data Engineering*

(ICDE), Boston, MA, pages 79–90, 2004.
16. G. I. Webb. Discovering significant patterns. *Machine Learning*, 68(1): 1–33, 2007.
17. G. I. Webb. Self-sufficient itemsets: An approach to screening potentially interesting associations between items. *ACM Transactions on Knowledge Discovery from Data*, 4(1), 2010.

A Proof of Proposition 2

In order to prove the proposition we need the following lemma which we will state without the proof.

Lemma 2 *Assume that a sequence $S = s_1, \ldots, s_n$ covers an episode G. If there is a source vertex v such that $s_1 = lab(v)$, then $s_2, \ldots, s_n$ covers $G \setminus v$. Otherwise, $s_2, \ldots, s_n$ covers G.*

Proof (of Proposition 2) We need to prove only "only if" case. Assume that $S = s_1, \ldots, s_n$ covers an episode G.

We will prove the proposition by induction over n. Obviously, the result holds for $n = 0$. Write $S' = s_2, \ldots, s_n$.

If there is no source vertex in G with a label s_1, then $gr(M, S) = gr(M, S')$. Now the lemma implies that S' covers G and the induction assumption implies that $gr(M, S') = G$.

If there is a source vertex v in G such that $lab(v) = s_1$, then $gr(M, S) = gr(M, S', G(v))$. Note that the $G(v)$ and its descendants form exactly $M(H)$, where $H = G \setminus v$. That is, $gr(M, S) = G$ if and only if $gr(M(H), S') = H$. The lemma implies that S' covers H and the induction assumption implies that $gr(M(H), S') = H$ which proves the proposition. □

B Proof of Proposition 5

In order to prove the proposition we need the following proposition, which essentially describes the properties of a log-likelihood of a log-linear model. The proof of this proposition can be found, for example, in [4].

Proposition 8 *Assume that we are given a set of k functions $T_i : \Omega \to \mathbb{R}$, mapping an object from some space Ω to a real number. For n real numbers, $r_1, \ldots, r_k$, define*

$$Z(r_1, \ldots, r_k) = \sum_{\omega \in \Omega} \exp \sum_{i=1}^{k} r_i T_i(\omega) \quad .$$

Define a distribution

$$p(\omega) = \frac{\exp \sum_{i=1}^{k} r_i T_i(\omega)}{Z(r_1, \ldots, r_k)} \quad .$$

Let X be a multiset of events from Ω. Define

$$c(r_1, \ldots, r_k) = \sum_{\omega \in X} \log p(\omega) \quad .$$

Then c is a concave function of $r_1, \ldots, r_k$. In fact

$$\frac{\partial c}{\partial r_i} = \sum_{\omega \in X} (T_i(\omega) - \mathrm{E}_p[T_i])$$

and

$$\frac{\partial c}{\partial r_i r_j} = |X|(\mathrm{E}_p[T_i]\,\mathrm{E}_p[T_j] - \mathrm{E}_p[T_i T_j]) \quad .$$

Proof (of Proposition 5) In order to prove the result we need to rearrange the terms in $\log p(\mathcal{S})$ based on current state. In order to do that, let us define L_H to be a multiset of labels that occur in $\mathcal{S}$ while the current state is H, that is,

$$L_H = \bigcup_{\substack{s_1,\ldots,s_n=S_i \\ i=1,\ldots,m}} \{s_j \mid gr(s_1,\ldots,s_{j-1}) = H\} \quad .$$

We can now rewrite the log-likelihood as

$$\log p(\mathcal{S}) = \sum_{H \in V(M)} \sum_{l \in L_H} \log p(l \mid H) \quad . \tag{2}$$

All we need to show now is that each term can be expressed in the form given in Proposition 8. In order to do that, define for each label l an indicator function

$$T_l(s) = \begin{cases} 1, & \text{if } l = s, \\ 0, & \text{otherwise} \quad . \end{cases}$$

Also, define indicator functions whether the transition is in C_1 or C_2, that is, define T_1 and T_2 as

$$T_i(s) = \begin{cases} 1, & \text{if there is } (H,F) \in C_i \text{ with } lab(H,F) = s, \\ 0, & \text{otherwise} \quad . \end{cases}$$

We have now

$$p(l \mid H) = \frac{1}{Z_H} \exp\left(t_1 T_1(l) + t_2 T_2(l) + \sum_{s \in \Sigma} u_s T_s(l) \right) \quad .$$

Since Z_H corresponds exactly to the normalization constant in Proposition 8, we have shown that

$$\sum_{l \in L_H} \log p(l \mid H)$$

is a concave function. The sum of concave functions is concave, proving the result. □

The proof also reveals how to compute the gradient and the Hessian matrix. These are needed if we are optimize $\log p(\mathcal{S})$. Since $\log p(\mathcal{S})$ is a sum of functions given in Proposition 8 the gradient and the Hessian matrix of $\log p(\mathcal{S})$ can be obtained by summing gradients and Hessian matrices of individual terms of Equation 2.

C Proof of Proposition 4

Proof Let $F = gr(M, s_1, \ldots, s_{n-1})$.

If $F = H$, then we remain in H only if s_n is not a label of an outgoing edge. The probability of this is equal to q.

If $F \neq H$, the only way $gr(M, S) = H$, is that F is a parent of H and the label connecting F to H is equal to s_n. This gives us the result. □

D Computing gradient descent

We use Newton-Raphson method to fit the model. In order to do this we need to compute the gradient and the Hessian matrix with respect to the parameters. This can be done efficiently as described by the following proposition.

Proposition 9 *Let G be an episode and let $M = M(G)$. Let H be a state in M.*

Let C_1 and C_2 be two disjoint subsets of $E(M)$. Define L_i to be the set of labels such that $l \in L_i$ if and only if there is an edge $(H, F) \in C_i$ labelled as l. Let J be a matrix of size $2 \times |\Sigma|$ such that $J_{il} = 1$ if $l \in L_i$, and 0 otherwise.

Let v be a vector of length $|\Sigma|$ such that $v_l = p(l \mid H)$ is equal to the probability of generating label l. Define $w = Jv$.

Let c be the count of how often we stay in H,

$$c = |\{(i,j) \mid s = S_j, H = gr(M, (s_1, \ldots, s_i))\}| \quad .$$

Let n be a vector of length $|\Sigma|$,

$$n_l = |\{(i,j) \mid s = S_j, H = gr(M, (s_1, \ldots, s_{i-1})), s_i = l\}|,$$

to contain the number of symbols labelled as l visited in $\mathcal{S}$ while being in the state H.

Let $V = diag(v)$ and let $W = diag(w)$. Define

$$d_H = \begin{bmatrix} n - cv \\ J(n - cv) \end{bmatrix} \quad \text{and} \quad B_H = c \begin{bmatrix} V - vv^T & VJ^T - vw^T \\ JV - wv^T & W - ww^T \end{bmatrix} \quad .$$

Then the gradient and hessian of $\log p(\mathcal{S})$ at $\{u_i\}$, t_1 and t_2 is equal to

$$d = \sum_{H \in V(M)} d_H \quad \text{and} \quad B = \sum_{H \in V(M)} B_H \quad .$$

Proof Proposition 8 and Proposition 5 imply that the gradient of $q_H = \sum_{l \in L_H} \log p(l \mid H)$ is equal to d_H and the hessian is equal to B_H. Since $\log p(\mathcal{S}) = \sum_{H \in V(M)} q_H$, the result follows. □

In order to obtain additional speed-ups, first notice that Proposition 9 implies that we do need to scan the original sequence set every time. Instead it is enough to compute the vector n and a scalar c for each state H. Moreover, for a fixed episode G, the rank does not depend on probabilities of individual labels that do not occur in G. In other words, we can treat all labels that do not occur in G as one label. This will reduce the length of the gradient and the size of the hessian from $|\Sigma| + 2$ to $|V(G)| + 3$, at most. These speed-ups make solving the model very fast in practice.