# Significance of Episodes Based on Minimal Windows

Nikolaj Tatti
*Advanced Database Research and Modelling (ADReM)*
*University of Antwerp, Antwerp, Belgium*
*nikolaj.tatti@gmail.com*

***Abstract*—Discovering episodes, frequent sets of events from a sequence has been an active field in pattern mining. Traditionally, a level-wise approach is used to discover all frequent episodes. While this technique is computationally feasible it may result in a vast number of patterns, especially when low thresholds are used.**

**In this paper we propose a new quality measure for episodes. We say that an episode is significant if the average length of its minimal windows deviates greatly when compared to the expected length according to the independence model. We can apply this measure as a post-pruning step to test whether the discovered frequent episodes are truly interesting and consequently to reduce the number of output.**

**As a main contribution we introduce a technique that allows us to compute the distribution of lengths of minimal windows using the independence model. Such a computation task is surpisingly complex and in order to solve it we compute the distribution iteratively starting from simple episodes and progressively moving towards the more complex ones. In our experiments we discover candidate episodes that have a sufficient amount of minimal windows and test each candidate for significance. The experimental results demonstrate that our approach finds significant episodes while ignoring uninteresting ones.**

***Keywords*-episode mining; statistical test; independence model; minimal window**

## I. Introduction

Discovering episodes, frequent patterns from an event sequence has been a fruitful and active field in pattern mining since their original introduction in [1]. Essentially an episode is a set of events that should occur close to each other with possibly some constraints on the order of the occurrences.

The most common way of defining a quality measure of an episode is the number of windows of fixed length in which the episode can be found. Such a measure is antimonotonic and hence all frequent episodes can be found using APRIORI approach given in [1]. This quality measure has two significant problems. First, the results will depend greatly on the length of the window. If the window is too small, then some interesting occurrences are ignored. On the other hand, if the window is too large, the behavior of occurrences in a single window is ignored.

**Example 1.** *Consider a serial episode $(a \rightarrow b)$ and two sequences 'abababababababab' and 'abacbadbaxbagbab'. If we fix the length of a window to be 6 (or larger), then the number of windows covering the episode will be the same for the both sequences. However, occurrences of the episode in these sequences are different.*

The second problem is that this measure has no way of incorporating any background knowledge. For example, assume that we know that event $a$ happens relatively seldom, then we are not surprised by the fact if we observe that an episode containing $a$ also occur seldom. Alternative approaches to deal with either the first problem or the second have been proposed and we discuss them in Section VIII.

In this paper we propose a new quality measure for the episodes. Our approach tackles simultaneously both aforementioned problems. To be more specific, given an episode $G$, we consider the lengths of minimal windows of $G$. To include background knowledge we assume that for each symbol we have a probability of its occurrence in the sequence. We then compute the expected length of the minimal window based on a model in which the symbols are independent of each other. We say that the episode is significant if the observed minimal windows have abnormal length, that is, the minimal windows are either too short or too long.

**Example 2.** *Assume that we have an alphabet of size 3, $\Sigma = \{a, b, c\}$. Assume that the probabilities for having a symbol are $p(a) = 1/2$, $p(b) = 1/4$, and $p(c) = 1/4$. Let $G$ be a serial episode $a \rightarrow b$. Then $s$ is a minimal window for $G$ if and only if it has a form $ac \cdots cb$. Hence the probability of a random sequence $s$ of length $k$ to be a minimal window for $G$ is equal to*

$$p(s \text{ is a minimal window of } G, |s| = k) = \frac{1}{2} \times \frac{1}{4} \times \frac{1}{4^{k-2}}.$$

*We are interested in a probability of a minimal window having length $k$. To get this we divide the joint probability by the probability $p(s \text{ is a minimal window of } G) = 1/6$. Using this normalization we get that the probability of a minimal window having length $k$ is equal to*

$$p(|s| = k \mid s \text{ is a minimal window of } G) = 3/4 \times 1/4^{k-2},$$

*for $k \geq 2$, and 0 otherwise. In this case the distribution is geometric and the expected length of a minimal window is then $7/3 \approx 2.3$.*

*On the other hand, assume that we have a sequence $s = accbabacb$. The minimal windows in $s$ are $s[1, 4]$, $s[5, 6]$, and*

$s[7,9]$. *Hence, the observed average length is* $(4{+}2{+}3)/3 = 3$.

Computing the probability of the length for a minimal window turns out to be a surprisingly complex problem. We attack this problem in Section IV by introducing a certain graph having episodes as the nodes. Then using this structure we are able to compute the probabilities inductively, starting from simple episodes and moving towards more complex ones.

Our recipe for the mining process is as follows: Given the sequence we first split the sequence in two. The first sequence is used for discovering candidate episodes, in our case episodes that have a large number of minimal windows (see Section VI for more details). Luckily, this condition is antimonotonic and we can mine these episodes using a standard APRIORI method. We also compute the needed probabilities for the events from the first sequence. Once the candidate episodes are discovered and the model is computed we compare the expected length of a minimal window against the average length of the observed minimal windows from the *second* sequence using a simple $Z$-test. This step allows us to prune uninteresting episodes, that is, the episodes that obey the independence model.

The rest of the paper is structured as follows. In Sections II–III we introduce the preliminary definitions and notation. In Section IV we lay out our approach for computing the independence model. We introduce our method for evaluating the difference between the observed windows and the independence model in Section V. We discuss mining candidate episodes in Section VI. Our experiments are given in Section VII. We present the related work in Section VIII and we conclude our work with discussion in Section IX.

## II. Preliminaries and Notation

We begin by presenting preliminary concepts and notations that will be used throughout the rest of the paper. In this section we will introduce the notions of sequence and episodes.

A *sequence* $s = (s_1, \ldots, s_L)$ is a string of symbols coming from a finite *alphabet* $\Sigma$, that is, we have $s_i \in \Sigma$. Such sequences are generated from random sources, hence we also treat $s$ as a random variable in our analysis. Given a sequence $s$ and two indices $i$ and $j$, such that $i \leq j$, we denote by $s[i,j] = (s_i, \ldots, s_j)$ a sub-sequence of $s$.

An episode $G$ is represented by an acyclic directed graph with labeled nodes, that is $G = (V, E, lab)$, where $V = (v_1, \ldots, v_K)$ is the set of nodes, $E$ is the set of directed edges, and $lab$ is the function $lab : V \to \Sigma$, mapping each node $v_i$ to its label.

Given a sequence $s$ and an episode $G$ we say that $s$ *covers* the episode if there is an *injective* map $f$ mapping each node $v_i$ to a valid index such that the node and the corresponding sequence element have the same label, $s_{f(v_i)} = lab(v_i)$, and that if there is an edge $(v_i, v_j) \in E$, then we must have $f(v_i) < f(v_j)$. In other words, the parents of the node $v_i$ must occur in $s$ before $v_i$. We define a binary function $c(s; G)$ such that $c(s; G) = 1$ if and only if $s$ covers $G$. Traditional episode mining is based on searching episodes that are covered by sufficiently many sub-windows of certain fixed size.

An elementary theorem says that in directed acyclic graph there exists a sink, a node with no outgoing edges. We denote the set of sinks by $sinks(G)$. Given an episode $G$ and a node $v$, we define $G - v$ to be the sub-episode obtained from $G$ by removing $v$ (and the incident edges).

Given a collection of episodes $\mathcal{G}$ we say that the collection is downward closed, if for a given $G \in \mathcal{G}$, each subgraph $H$ of $G$ is included in $\mathcal{G}$. Note that the empty episode is included in $\mathcal{G}$. Throughout the whole paper we will be working with downward closed collections of episodes $\mathcal{G}$.

## III. Minimal Windows of Episodes

Episode mining is based on finding episodes that occur often enough in sliding window. We approach the problem from a different angle. Given a sequence and a candidate episode we first discover the set of all minimal windows in which the given episode occurs. Once we have obtained this set we will study the length of these windows. If their distribution is abnormal, either the lengths are too short, or too long, we consider that we have discovered an important episode.

In order to make the preceding discussion more formal, let $G$ be an episode, and let $s$ be a sequence. We say that $s$ is a *minimal window* for $G$ if $G$ is covered by $s$ but not by any proper sub-window of $s$. We define a function $m(s; G)$ returning a binary value. The function $m(s; G) = 1$ if and only if $s$ is a minimal window starting for $G$.

Let $s$ be a random sequence of length $L$, and write $t = s[1, L-1]$ and $u = s[2, L]$. Note that we can write the probability of $s$ being a minimal window as

$$\begin{aligned} p(m(s;G)) &= p(c(s;G)) - p(c(t;G) \vee c(u;G)) \\ &= p(c(s;G)) - p(c(t;G)) - p(c(u;G)) \\ &\quad + p(c(t;G), c(u;G)) . \end{aligned} \tag{1}$$

Our main focus is the distribution of lengths of minimal windows, that is, we are interested in

$$p_G(k) = p(|s| = k \mid m(s;G) = 1) ,$$

where $s$ is a random sequence. Note that this distribution is defined for all $k$. In practice, we compute $p(m(s;G))$ for $k = 1, \ldots, K$, where $K$ is some suitable predetermined constant. Once these values are computed we normalize them so that $p_G(k)$ becomes a proper distribution. This is equivalent to saying that we are not interested in minimal windows whose length exceeds $K$. From now on $K$ will always denote the maximal length of a minimal window.

We should point out that even though we limit ourselves to windows of maximal size $K$ this limitation is not as severe as using windows of fixed size $K$. While we ignore information of longer windows we still are able to detect any deviation occurring with the length of minimal windows shorter or equal than $K$. On the other hand, in the fixed window approach the information of the length of minimal window is discarded as long as it is short enough.

## IV. Computing the Minimal Windows from Independence Model

We devote this section for computing the distribution of lengths of minimal windows. That is we are given probabilities $p(a)$ for each symbol in an alphabet and a set of episodes $\mathcal{G}$ and for each $G \in \mathcal{G}$ we wish to compute $p_G(k)$, the probability of a minimal window of $G$ being of size $k$, according to the independence model.

Our approach for calculating minimal windows is based on Eq. 1. According to this equation we need to solve the probabilities $p(c(s;G))$ and $p(c(s[1,L-1];G), c(s[2,L];G))$. We will achieve this by building certain finite state machines where the states will correspond to the episodes.

### A. Episode Set as Finite State Machine

It will be fruitful to represent the given set of episodes $\mathcal{G}$ as a certain finite state machine. To be more precise, we define a finite state machine as a DAG $M_\mathcal{G} = (V(M_\mathcal{G}), E(M_\mathcal{G}))$. The states $V(M_\mathcal{G})$ are exactly the episodes $\mathcal{G}$. Let $X, Y \in \mathcal{G}$ be two episodes and let $v$ and $w$ be the corresponding states. An edge $e = (v,w) \in E(M_\mathcal{G})$ with a label $a = lab(e)$ exists if and only if $X = Y - n$, where $n$ is a sink node of $Y$ labeled as $a$. In other words, there should be an edge between an episode and an episode obtained by removing a sink node with a label $a$.

**Example 3.** *We consider a downward set of closed episodes $\mathcal{G} = \{(a, b \to a), (b \to a), (a,b), (a), (b), (a,a), \emptyset\}$. The machine $M_\mathcal{G}$ is given in Figure 1.*

Given a state $v$ in $M_\mathcal{G}$ we say that $s$ *covers* $v$ if there is a sequence $t = \{s_{i_1}, \ldots, s_{i_N}\}$ such that $v$ can be reached when $t$ is given as an input. In that case we set $c(s;v) = 1$, and 0 otherwise. Similarly we define $m(s;v)$, when $s$ covers $v$ but $s[2,L]$ and $s[1,L-1]$ does not cover $v$.

Comparing this to the definition of coverage for the episode we see the immediate result.

**Proposition 4.** *The sequence $s$ covers an episode $G \in \mathcal{G}$ if and only if $s$ covers the corresponding $v$ in $M_\mathcal{G}$. Consequently, a minimal window of $G$ is equivalent to the minimal window of $v$.*

Let $M$ be a finite state machine and let $v$ be a state in $M$ We say that $v$ is *monotonic* if a sequence $s$ covering $v$ also covers any parent state of $v$. If every state in $M$ is monotonic we say that $M$ is *monotonic*.

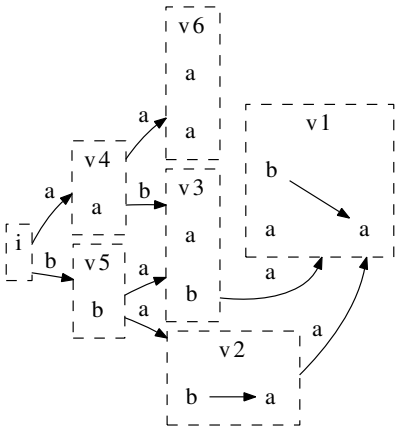


Figure 1. The machine $M_\mathcal{G}$ for the episodes $\mathcal{G} = \{(a, b \to a), (b \to a), (a,b), (a), (b), (a,a), \emptyset\}$. Each edge represents a removed sink between the episode and the parent episode.

A direct corollary of Proposition 4 states that $M_\mathcal{G}$ is monotonic.

**Lemma 5.** *The machine $M_\mathcal{G}$ induced from $\mathcal{G}$ is monotonic.*

*Proof:* Let $v$ a state in $M_\mathcal{G}$ and let $w$ be its parent state. Let $X$ be the episode represented by the state $v$ and let $Y$ be the episode represented by the state $w$. If $v$ covers $s$, then it must cover $X$. Since $Y$ is an episode obtained from $X$ by removing one sink, $s$ also covers $Y$ and thus cover $w$. ■

### B. Computing Coverage for States of Simple Machines

In this section we will demonstrate how to compute the probabilities $p(c(s;v))$ for the state $v$ in $M$. We will make some simplifying assumption concerning the structure of $M$, and then in the next section we demonstrate how this limitations can be removed.

We say that a machine $M$ is *simple* if incoming edges for each state $v$ in $M$ have unique labels. Generally, the episode machine $M_\mathcal{G}$ is not simple. However, if the episodes $\mathcal{G}$ have only nodes with unique labels, then $M_\mathcal{G}$ will be simple.

Our approach is to compute the coverage of a state $v$ based on the coverage of its parent state.

**Proposition 6.** *Let $M$ be simple and monotonic. Let $v$ be a state in $M$, and let $s$ be a random sequence of length $L$ with independent symbols. Let $t = s[1, L-1]$ be the subsequence of $s$ without the last element. Define probability $d$ as $p(c(s;v)) = d + p(c(t;v))$. Then*

$$d = \sum_{e=(w,v)\in E(M)} p(lab(e)) \left(p(c(t;w)) - p(c(t;v))\right).$$

*Proof:* By definition, we have $d = p(c(s;v)) - p(c(t;v))$, that is, $d$ is the probability of sequence $s$ covering

$v$ but $t = s[1, L-1]$ not covering it. Assume that $s$ is such sequence. This implies that there is an edge $e = (w, v)$ with $lab(e) = s_L$. Fix $s_L = lab(e)$. Since $M$ is simple the only path to reach $v$ must use the unique $e$. We also must have that $t$ covers $w$ but not $v$. Note that this probability can be written as

$$p(c(t;w)) - p(c(t;w), c(t;v)) = p(c(t;w)) - p(c(t;v)),$$

where the equality follows from since $M$ is monotonic.

The probability of $s_L$ being $lab(e)$ is $p(lab(e))$. The result follows by combining these probabilities. ■

Let us abuse the notation and write $p(c(L;v))$ to mean the probability $p(c(s;v))$ where $s$ is a random sequence of length $L$. The proposition gives us means to compute $p(c(L;v))$ in an iterative fashion from $p(c(L-1;v))$ and from the coverage of parent state. The algorithm for computing the coverage is given in Algorithm 2.

**Algorithm 1** Recursive sub-procedure COVERSTATE for computing the coverage of state $v$.

```
for e = (w, v) ∈ E(M) do
    if coverage for w is not computed then
        CoverState(w).
    end if
end for
for k = 1, ... K. do
    d ← 0.
    for e = (w, v) ∈ E(M) do
        x ← p(c(k − 1; w)) − p(c(k − 1; v)).
        d ← d + p(lab(e)) x.
    end for
    p(c(k; v)) ← d + p(c(k − 1; v)).
end for
```

**Algorithm 2** Algorithm COVER for computing the coverage of state for a simple and monotonic machine $M$.

```
s ← the source state of M.
p(c(k; s)) ← 1, for k = 0, ..., K.
for v sink state in M do
    CoverState(v).
end for
```

To analyze the computational complexity, we first note that computing the coverage of state $v$ requires $O(KL)$ steps where $L$ is the number of incoming edges of $v$. Thus computing the coverage of the complete graph will require $O(|E(M)|K)$ steps. However, in practice the process is more complex. The computations are not numerically stable due to rounding errors in floating-point numbers. To solve this problem we have to resort to exact rational numbers. Using such numbers implies that simple computations are no longer unit operations making the computation times longer.

### C. Transforming non-simple Machines

In order to use Algorithm 2 our state machine needs to be simple and monotonic. The machine $M_{\mathcal{G}}$ is monotonic but not simple. Luckily, we can define a new simple and monotonic machine from which we can compute the coverage. Informally, if we reverse the direction of the edges in $M_{\mathcal{G}}$, then making the machine simple is equal to making the reversed non-deterministic machine deterministic.

In order to make this formal, let us first give some definitions. Let $V$ be a subset of states in $M$. Let $a$ be a label. We also define

$$sub(V;a) = \{w \mid e = (w, v) \in E(M), lab(e) = a, v \in V\}$$

to be the set of parents of each $v \in V$ connected with an edge having a label $a$. We define

$$par(V;a) = \min(sub(V;a) \cup V),$$

where $\min(X)$ results in minimal states of $X$ with respect to the parenthood in $M$. This guarantees that $par(V;a)$ contains no state $v, w$ such that $v$ is an ancestor of $w$. We also need to define

$$in(V) = \{lab(e) \mid e = (w, v) \in E(M), v \in V\}$$

to be the set of labels of all incoming edges. We define the closure of $V$ inductively to be the collection of sets of states

$$cl(V) = \{V\} \cup \bigcup_{a \in in(V)} cl(par(V;a))$$

and $cl(V) = \{V\}$ if $in(V)$ is empty.

We are now ready to transform $M$ into a simple machine, which we denote by $sm(M)$. The states of the machine $sm(M)$ are

$$V(sm(M)) = \bigcup_{v \in V(M)} cl(\{v\}).$$

Let $V$ be a state in $sm(M)$ and let $a \in in(V)$. Let $W = par(V;a)$. Note that this state exists in $sm(M)$. We define an edge labeled as $a$ from $W$ to $V$. Let $i$ be the initial state in $M$. For $M_{\mathcal{G}}$ it is the state corresponding to the empty episode. Then $\{i\}$ is the initial state for $sm(M)$. From now on $\{i\}$ will always denote the initial state of $sm(M)$.

**Example 7.** *We continue Example 3. Note that $M_{\mathcal{G}}$ is not simple since $v_1$ has two incoming edges with a label a. The transformed machine $sm(M_{\mathcal{G}})$ is given in Figure 2. Note that, in addition to the states already existing in $M_{\mathcal{G}}$ is has now two extra states, namely $\{v_2, v_3\}$ and $\{v_2, v_4\}$. Also note that $sm(M_{\mathcal{G}})$ is simple.*

It is obvious that $sm(M)$ is simple. The following proposition reveals the expected relationship between $M$ and $sm(M)$.

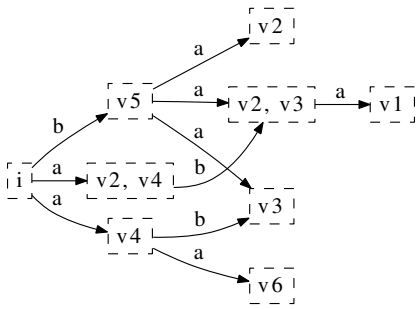


Figure 2. The transformed machine $sm(M_{\mathcal{G}})$ of the machine $M_{\mathcal{G}}$ given in Figure 1.

**Proposition 8.** *Let $M$ be a monotonic machine. Let $V = \{v_1, \ldots, v_N\}$ be the state in $sm(M)$. Then a sequence $s$ covers $V$ if and only if $s$ covers at least one $v_i$.*

*Proof:* We will prove this by induction. Assume that the proposition holds for all parent states of $V$.

Assume that $s$ covers $V$. Let $t$ be a sub-sequence of $s$ that leads $sm(M)$ from the source state $\{i\}$ to $V$. Let $s_e$ be the last symbol of $s$ occurring in $t$. There is a parent state $W = \{w_1, \ldots, w_L\}$ which $s[1, e-1]$ covers. By the induction assumption at least one $w_k$ is covered by $s[1, e-1]$. If there is $v_j = w_k$, then $v_j$ is covered by $s$, otherwise there is $v_j$ that has $w_k$ as a parent state. The edge connecting $v_j$ and $w_k$ has a label $s_e$. Hence $s$ covers $v_j$ also.

To prove the other direction assume that $s$ covers $v_j$. Let $t$ be a sub-sequence that leads $M$ from the source state to $v_j$. Let $s_e$ be the last symbol occurring in $t$. Let $w$ be the parent state of $v_j$ connected by an edge with a label $s_e$. Since $s_e \in in(V)$, we must have $W$ as a parent state of $V$ such that either $w \in W$ or an ancestor state $u$ of $w$ is in $W$. In the latter case, since $M$ is monotonic, $s$ covers $u$. In either case, by the induction assumption $s[1, e-1]$ covers $W$. Hence $s$ covers $V$. ∎

**Corollary 9.** *Sequence $s$ covers $v$ in $M$ if and only if $s$ covers $\{v\}$ in $sm(M)$.*

**Corollary 10.** *Let $M$ be a monotonic machine, then $sm(M)$ is also monotonic.*

*Proof:* Let $V$ be a state in $sm(M)$ and let $s$ cover $V$. Let $W$ be a parent state of $V$. Let $v \in V$ such that $s$ covers $v$. If $v \in W$, then $s$ covers $W$. Otherwise there is an ancestor state $w \in W$ of $v$. Since $M$ is monotonic, $s$ covers $w$ and thus $W$. ∎

The corollaries give us means to compute the coverage of states in $M_{\mathcal{G}}$ by solving the coverage of the states $sm(M_{\mathcal{G}})$ using Algorithm 2.

### D. Computing Co-coverage

Our last challenge is to compute the term $p(c(t; G), c(u; G))$ in Eq. 1. In order to do that we design a special finite state machine, denoted by $co(M)$, in which the coverage of certain states will correspond to the last term in Eq. 1. The construction of this machine is based on the previous machines $M = M_{\mathcal{G}}$ and $sm(M)$. To avoid confusion we use $v$ and $w$ for the states in $M$, $V$ and $W$ for the states in $sm(M)$, and greek letters $\alpha, \beta, \ldots$ for the states in $co(M)$. We proceed by constructing the machine first and then prove that it gives us the desired probabilities.

There are three different kinds of states in $co(M)$. The first group consists of one state, namely $\eta = \{i\}$, where $\{i\}$ is the initial state of $sm(M)$. This state will be the initial state of $co(M)$.

The second group consists of certain pairs of states from $sm(M)$. Let $V$ and $W$ be states in $sm(M)$ and write $\alpha = (V, W)$. The machine will be constructed in such manner that $s$ will cover $\alpha$ in $co(M)$ if and only if $s$ covers $V$ and $s[2 : L]$ covers $W$ in $sm(M)$. In order to achieve this we first define a closure

$$cl(\alpha) = \{\alpha\} \cup \bigcup_{a \in in(V) \cup in(W)} cl((par(V; a), par(W; a))).$$

Let $v$ be a state in $M$ that is not the source state. For each label $a \in in(\{v\})$ we add the states from $cl((\{v\}, par(\{v\}; a)))$ into $co(M)$. In addition, we add the states from $cl((\{v\}, \{v\}))$. We add an edge with a label $a$ to $\alpha = (V_1, V_2)$ from $\beta = (W_1, W_2)$ if $W_k = par(V_k; a)$ for $k = 1, 2$ with one exception: if $par(V_1; a) = V_2 = \{i\}$, then instead if connecting $\alpha$ to $(\{i\}, \{i\})$ we connect $\alpha$ to $\eta = \{i\}$. We also connect the state $(\{i\}, \{i\})$ to $\eta$ with an edge accepting any symbol from the alphabet. See Example 11 for illustration.

We will now define our last group of states. For any state $v$ that is not a source in $M$ we add a state $\alpha = v$. For each $a \in in(\{v\})$, we add an edge with a label $a$ from $(\{v\}, par(\{v\}; a))$ to $\alpha$. We also add an edge from $(\{v\}, \{v\})$ to $\alpha$ accepting any symbol outside $in(\{v\})$.

**Example 11.** *We continue the toy example given in Example 7. A part of the machine $co(M_{\mathcal{G}})$ is given in Figure 3. Namely we show the machine solving the co-coverage for the episodes $v_2 = (b \to a)$ and $v_5 = (b)$.*

Now that we have defined our machine we are ready to prove that the coverage of the states of the last group actually corresponds to the last term in Eq 1.

First, we need to point out a certain property of $sm(M)$.

**Lemma 12.** *Let $M$ be a monotonic machine. Let $s$ be a sequence of length $L$ covering a state $V$ in $sm(M)$. Then $s[1, L-1]$ covers $par(V; s_L)$.*

*Proof:* If $s_L \notin in(V)$, then a sub-sequence $t$ leading to $V$ does not contain $s_L$. Hence $s[1, L-1]$ covers $V = par(V; s_L)$. Assume that $s_L \in in(V)$. Let $W = par(V; s_L)$. If $s[1, L-1]$ covers $V$, then by the monotonicity

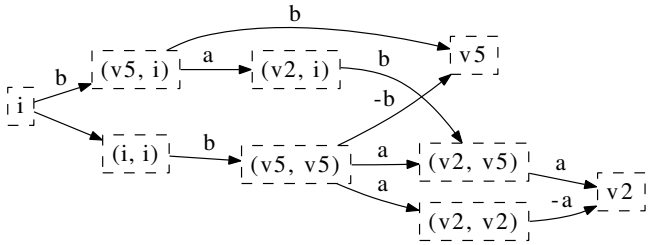


Figure 3. A part of $co(M_{\mathcal{G}})$. Here we have only included states corresponding to the episodes $(b)$ and $(b \rightarrow a)$. The edge $-a$ means that the edge accepts any symbol but $a$.

of $sm(M)$, $s$ also covers $W$. If $s[1, L-1]$ does not cover $V$, then $t$ must have $s_L$ as a last symbol and since $sm(M)$ is simple, $t$ must go through $W$ Hence, $s[1, L-1]$ covers $W$. ■

Our second step is to describe the coverage of the intermediate states in $co(M)$.

**Proposition 13.** *Let $s$ be a sequence of length $L$. Let $\alpha = (V_1, V_2)$ be a state in $co(M)$, then $s$ covers $\alpha$ if and only if $s$ covers $V_1$ and $s[2, L]$ covers $V_2$.*

*Proof:* We will prove the result by induction. To prove the first step assume that $par(V_1; a) = par(V_2; a) = \{i\}$. If $V_2 = \{i\}$, then $V_1$ is connected to the state $\eta = \{i\}$ (in $sm(M)$) by an edge with a label $a$. Hence, $s$ covering $\alpha$ is equivalent to $s$ covering $V_1$. The result follows since the state $V_2 = \{i\}$ is automatically covered. Assume now that $V_2 \neq \{i\}$. In this case $s[2, L]$ covering $V_2$ implies that $s[2, L]$ (and hence also $s$) covers $V_1$. Since $\alpha$ is connected to $(\{i\}, \{i\})$, $s$ covering $\alpha$ is equivalent that $s[2, L]$ has a symbol $a$. But this is equivalent for $s[2, L]$ covering $V_2$.

Assume now that the result holds for all parent states of $\alpha$. Assume that $s$ covers $\alpha$. There must be a symbol $s_e$ and a parent state $\beta = (W_1, W_2)$ linked to $\alpha$ by an edge with a label $s_e$ such that $s[1, e-1]$ covers $\beta$. By the assumption $s[1, e-1]$ covers $W_1$ and $s[2, e-1]$ covers $W_2$. Thus, $s[1, e]$ covers $V_1$ and $s[2, e]$ covers $V_2$.

Assume now that $s$ covers $V_1$ and $s[2, L]$ covers $V_2$. Let $s_e$ be the last element in $s$ such that $s_e \in in(V_1)$ or $s_e \in in(V_2)$. By Lemma 12 we must have that $s[1, e-1]$ covers $par(V_1; s_e) = W_1$ and $s[2, e-1]$ covers $par(V_2; s_e) = W_2$. Hence by the induction assumption $s[1, e-1]$ covers $\beta = (W_1, W_2)$ and consequently, since $\beta$ is a parent state of $\alpha$, $s[1, e]$ covers $\alpha$. ■

Now we are ready to prove that the coverage of states in the last group is exactly what we wish to have.

**Proposition 14.** *Let $\alpha$ be a state in $co(M)$ corresponding to a state $v$ in $M$. Then a sequence of length $L$ covers $\alpha$ if and only if $s[1, L-1]$ and $s[2, L]$ cover $v$.*

*Proof:* Assume that $s$ covers $\alpha$. Let $t$ be a sub-sequence of $s$ that leads to $\alpha$ from the source state. Let $s_e$ be a last symbol of $t$. If $s_e \in in(\{v\})$ then $t$ travels through $\beta = (\{v\}, par(\{v\}; s_e))$. By Proposition 13, $s[1, e-1]$ covers $\{v\}$ and $s[2, e-1]$ covers $par(\{v\}; s_e)$ and hence $s[2, e]$ covers $v$. If $s_e \notin in(\{v\})$, then $t$ travels through $(\{v\}, \{v\})$ and result follows again from Proposition 13.

To prove the other direction assume now that $s[1, L-1]$ and $s[2, L]$ cover $\{v\}$. Assume that $s_L$ is not in $in(\{v\})$. This implies that $s[2, L-1]$ covers $\{v\}$. Thus, by Proposition 13, $s[1, L-1]$ covers $(\{v\}, \{v\})$ and consequently $s$ covers $\alpha$. On the other hand, if $s_L \in in(\{v\})$, then Lemma 12 implies that $s[2, L-1]$ covers $par(\{v\}; s_L)$ and hence must cover, by Proposition 13, $(\{v\}, par(\{v\}; s_L))$. Consequently $s$ covers $\alpha$. ■

It is easy to see that $co(M)$ is simple and monotonic, if $M$ is monotonic. Hence, we can compute the coverage of $co(M_{\mathcal{G}})$ using Algorithm 2.

We can now compute the probability of $s$ being a minimal window using Eq. 1. First we solve the coverage using $sm(M_{\mathcal{G}})$. Secondly, we compute the co-coverage using $co(M_{\mathcal{G}})$. Once these are computed we can use Eq. 1.

Let us finish by discussing the relative sizes of the machines. It can be shown that the number of states in $sm(M_{\mathcal{G}})$ can be substantially larger than the number of states $M_{\mathcal{G}}$. Consequently, in the worst case our method is not polynomial. Such an explosion, however, requires a specific episode with many nodes having the same label. Such episodes are unlikely to be candidates if we are dealing with sequences that have a large alphabet distributed more or less evenly. Moreover, we demonstrate later that in our experiments the size of $sm(M_{\mathcal{G}})$ is about the same as the size of $M_{\mathcal{G}}$.

## V. Testing Minimal Windows

In this section we will describe how we test whether the discovered minimal windows obey the independence model.

We say that the episode is significant if the average length of the minimal windows is abnormally small or large. In order to measure the abnormality we will use a $Z$-test. In order to perform this test we need to show that the average length is asymptotically normal and compute the mean and the variance according to the independence model. This is not trivial since the minimal windows correlate within a single sequence $s$. For example, assume that we are looking for the parallel episode $(a, a, a)$, and assume that we have found a minimal window of size 3, that is, the window is $aaa$. Then the next minimal window will have a higher probability of being short, since we already have two $a$s. Thus the occurrences of minimal windows in $s$ are not independent even if $s$ obeys the independence model.

Assume that we are given a long random sequence $s$ of length $N$ and write $X_i$ to be a boolean random variable such that $X_i = 1$ if there is a minimal window starting at $i$th symbol. Also let $Y_i$ be the length of that minimal window

and 0 if there is no window. Note that the estimator of the average length is

$$M = \sum_i Y_i / \sum_i X_i.$$

Let us first show that $M$ is normally distributed. To see that note that $(Y_i, X_i)$ and $(Y_{i+K}, X_{i+K})$, where $K$ is the maximum length of a window, are independent. Hence, $(X_i, Y_i)$ is a *strongly mixing* sequence which allows us to use a Central Limit Theorem for dependent variables (given in [2], for example) so that $(\sqrt{N}X_i, \sqrt{N}Y_i)$ is asymptotically normal as $N$ approaches infinity. Let us denote by $C$ the covariance matrix of this limit distribution. Also write $p = \mathrm{E}[X_1]$ and $q = \mathrm{E}[Y_1]$. Using the same theorem we know that the components of $C$ are

$$\begin{aligned}
C_{11} &= \mathrm{E}\left[(Y_1 - q)^2\right] + 2\sum_{i=2}^{K} \mathrm{E}\left[(Y_1 - q)(Y_i - q)\right] \\
C_{22} &= \mathrm{E}\left[(X_1 - p)^2\right] + 2\sum_{i=2}^{K} \mathrm{E}\left[(X_1 - p)(X_i - p)\right] \\
C_{12} &= \mathrm{E}\left[(X_1 - p)(Y_1 - q)\right] + \\
&\quad \sum_{i=2}^{K} \mathrm{E}\left[(X_1 - p)(Y_i - q) + (Y_1 - q)(X_i - p)\right].
\end{aligned}$$

Let us define $m = q/p$ which is the average length of the minimal window. Since $f(y, x) = y/x$ is continuous and differentiable function at $(q, p)$, we know from Theorem 3.1 in [3] that $\sqrt{N}f(\sum Y_i, \sum_i X_i)$ is asymptotically normal with mean $m$ and variance $\sigma^2$, where

$$\sigma^2 = \nabla f^T C \nabla f = \frac{1}{p^2}\left(C_{22} - 2mC_{12} + m^2 C_{11}\right),$$

where $\nabla f = (1/p, -m/p)$ is the gradient of $f$ at $(q, p)$.

Thus in order to perform a statistical test for an episode $G$ given a sequence $s$, let $W$ be the sum of lengths of the discovered minimal windows. We consider the following statistic

$$Z = \frac{W - Nm}{\sqrt{N}\sigma}. \tag{2}$$

Based on the above discussion if $s$ truly comes from the independence model, then $Z$ is asymptotically distributed as a standard normal distribution $N(0, 1)$.

Our remaining task is to compute $m$ and $\sigma$. Note that since we know the probability of $s$ being a minimal window, we can compute $p$, $q$, $m$, and the first terms of $C_{11}$, $C_{12}$, and $C_{22}$. However the last terms of $C$ cannot be computed easily. We resolve this issue by simulating a sequence of independence model and estimating these terms from that sequence.

## VI. Mining Candidate Episodes with non-overlapping Minimal Windows

So far we have assumed that we already know what episodes we wish to test. In this section we will focus on mining candidate episodes.

In order to have a reliable $Z$-statistic (see Eq. 2), we need to have a decent number of minimal windows. Hence, a good criterion for a candidate episode is that the number of minimal windows exceeds some given threshold. This is the criterion used in MINEPI (see [1]). However, this condition is not antimonotonic as demonstrated in the next toy example.

**Example 15.** *Consider the sequence 'aba'. There are* 2 *minimal windows for the parallel episode* $(a, b)$*, yet there is only one minimal window for the episode* $(b)$*.*

We remedy this problem by making a stronger requirement. We search all the episodes whose number of non-overlapping windows exceed some given threshold. It turns out, that this condition is antimonotonic and we can search the episodes in a level-wise fashion.

Since there are several ways of selecting non-overlapping subcollection of minimal windows, we will give a more precise definition. Let $W$ be a sequence $W$ of minimal windows of an episode $G$ in a sequence $s$. Assume that the minimal windows in $W$ are ordered by their occurrences in $s$. We select the first window and remove any window that overlaps with the selected window. We repeat this until the $W$ has no more windows. We define $nm(G; s)$ to be the minimal windows discovered in such fashion. We first show that this approach produces the maximal number of samples.

**Proposition 16.** *Let* $V$ *be a collection of non-overlapping minimal windows of an episode* $G$ *in a sequence* $s$*. Then* $|V| \leq |nm(G; s)|$.

*Proof:* Let $W$ be the collection of possibly overlapping minimal windows of $G$ in $s$. We will prove that among any sub-collection of $W$ of non-overlapping windows, the collection $nm(G; s)$ has the maximal size. We will prove this by induction over the size of $W$.

Let $w \in W$ be the first window in $W$. Let $X$ be the set of windows that overlap with $w$ (note that $w \in X$). By the definition, we have $w \in nm(G; s)$, and the next window will be the first window outside $X$.

If $V \cap X = \emptyset$, then $V \subseteq W - X$, and the result follows from the induction assumption. Assume that $V \cap X \neq \emptyset$. Any two windows $x, y \in X$ must overlap, hence $V$ can contain exactly one member of $X$, say $x$. This means that $V - \{x\} \subseteq W - X$, and the result follows from the induction assumption. ∎

**Corollary 17.** *The quantity* $|nm(G; s)|$ *is antimonotonic.*

*Proof:* Let $H$ be a sub-episode of $G$. Then any minimal window in $nm(G; s)$ also contains a minimal window of

$H$. Let $V$ be a collection of minimal windows of $H$ constructed by taking one minimal window from each window $w \in nm(G; s)$. It is obvious that the windows in $V$ do not overlap and that $|V| = nm(G; s)$. Proposition 16 implies that $|nm(H; s)| \geq |V|$.

■

In [4] the authors introduce a measure for the episodes to be the maximal number of non-overlapping occurrences of the episode $s$. Since each occurrence is either a minimal window or contains a minimal window, Proposition 16 tells us that $nm(G; s)$ is exactly this measure.

## VII. Experiments

In this section we present our experiments with the quality measure using synthetic and real-world text sequences.

### A. Datasets

We conducted our experiments with several synthetic and real-world sequences.

The first synthetic sequence, *gen-ind* consisted of 200000 digits drawn independently from the uniform model. The purpose of this dataset is to show that our method finds very few significant episodes. The second synthetic sequence, *gen-co* also consisted of 200000 digits. The sequence was generated as follows. First we choose, by a fair coin flip, whether to generate a digit from $0 - 4$ or $5 - 9$. In the former case the digit was selected from a uniform model. In the latter case the probability of selecting the digit $i$ was proportional to $0.5^x$, where $x$ is the distance between the current location and the last location of the digit $i - 5$. Thus in this sequence, the digits $i$ and $i - 5$ tend to be close to each other.

Our third dataset, *moby*, was the novel Moby Dick by Herman Melville.[1] Our fourth sequence, *abstract* consisted of 739 first NSF award abstracts from 1990.[2] Our final dataset, *address*, consisted of inaugural addresses of the presidents of the United States.[3] To avoid the historic concept drift we entwined the speeches by first taking the odd ones and then even ones. The sequences were processed using the Porter Stemmer and the stop words were removed.

### B. Experimental Setup

Our experimental setup mimics the framework setup in [5] in which the data is divided into two parts, the first part is used for discovering the patterns and the second part for testing whether the discovered patterns were significant. We divided each sequence into two parts of equivalent lengths. We used the first sequence for discovering the candidate episodes and training the independence model. Then the discovered episodes were tested against the model using the second sequence.

[1]The book was taken from http://www.gutenberg.org/etext/15.

[2]The abstracts were taken from http://kdd.ics.uci.edu/databases/nsfabs/nsfawards.html

[3]The addresses were taken from http://www.bartleby.com/124/pres68.

| Sequence | Size | $\|\Sigma\|$ | $N$ | $K$ |
|---|---|---|---|---|
| *gen-ind* | 200000 | 10 | 4000 | 40 |
| *gen-co* | 200000 | 10 | 3500 | 35 |
| *moby* | 105719 | 10277 | 20 | 10 |
| *abstract* | 67828 | 6718 | 22 | 10 |
| *address* | 62066 | 5295 | 20 | 10 |

Table I
CHARACTERISTICS OF THE SEQUENCES AND THE THRESHOLD VALUES USED FOR MINING CANDIDATE EPISODES. THE SECOND COLUMN IS THE NUMBER OF SYMBOLS IN THE SEQUENCE. THE THIRD COLUMN IS THE THRESHOLD FOR THE NUMBER OF MINIMAL WINDOWS AND THE FOURTH COLUMN IS THE LARGEST MINIMAL WINDOW CONSIDERED.

As candidate episodes we considered only those episodes whose number of non-overlapping windows exceeded some threshold $N$. When computing the independence model and discovering minimal windows from the test data we only considered the minimal windows of at most $K$. We used $K = 10$ for the text sequences and $K = 35, 40$ for the synthetic sequences. These thresholds are given in Table I.

Let $\mathcal{G}$ be the set of candidates. Since we compute the samples from the test sequence, it is not guaranteed that an episode $G \in \mathcal{G}$ will have enough minimal windows. Hence we discard any episode whose number of minimal windows in the test sequence does not exceed $N$. We also remove any episodes having the variance 0 since for these episodes the minimal window will always be of the same known size. This set includes all singletons. Let us denote this set of episodes by $\mathcal{H}$. The sizes of these families along with the sizes of the machines $sm(M_{\mathcal{H}})$ and $co(M_{\mathcal{H}})$ are given in Table II.

For each episode $H \in \mathcal{H}$ we computed the $Z$-statistic given in Eq. 2. This value is asymptotically distributed as a standard normal distribution. We considered two different $P$-values. First, we computed a one-sided $P$-value to we examine whether $Z$ is abnormally small, thus our test will return small $P$-values if the minimal windows are significantly smaller than expected. Secondly, we computed a two sided $P$-value to test whether the average of lengths of minimal windows is significantly smaller or larger. The correlation between the minimal windows (see Section V) was computed by simulating a sequence with $10^6$ symbols. The computation of $P$-values lasted about 5 minutes for the generated sequences and less than a minute for text sequences. The most expensive step was the computation of the correlation terms explained in Section V.

### C. Significant Episodes

In this section we will focus on the episodes discovered by our approach.

From each candidate set we computed the significant episodes based on their $P$-values. As a significance level

| Sequence | $\|\mathcal{G}\|$ | $\|\mathcal{H}\|$ | $\|sm(M_\mathcal{G})\|$ | $\|co(M_\mathcal{G})\|$ |
|---|---|---|---|---|
| *gen-ind* | 4882 | 4872 | 4889 | 28046 |
| *gen-co* | 3993 | 3982 | 4221 | 24035 |
| *moby* | 724 | 137 | 726 | 2382 |
| *abstract* | 14569 | 116 | 14985 | 106901 |
| *address* | 482 | 78 | 483 | 1551 |

Table II
SIZES OF DATA STRUCTURES IN EXPERIMENTS. THE FIRST COLUMN IS THE NUMBER OF CANDIDATE EPISODES, THE SECOND COLUMN IS THE NUMBER OF EPISODES ACTUALLY TESTED. THE THIRD COLUMN IS THE NUMBER OF STATES IN $sm(M_\mathcal{G})$ AND THE FOURTH COLUMN IS THE NUMBER OF STATES IN $co(M_\mathcal{G})$.

we used $0.05$. We compared raw $P$-values and also adjusted $P$-values. The adjustment was done using the Benjamini Hochberg Procedure in order to control the FDR family-wise error [6]. The results are given in Table III.

| | Raw | | Adjusted | |
|---|---|---|---|---|
| Sequence | one-s. | two-s. | one-s. | two-s. |
| *gen-ind* | 446 | 355 | 0 | 0 |
| *gen-co* | 237 | 101 | 242 | 90 |
| *moby* | 23 | 20 | 12 | 9 |
| *abstract* | 41 | 42 | 15 | 15 |
| *address* | 20 | 19 | 3 | 3 |

Table III
SIGNIFICANT EPISODES ACCORDING TO THEIR RAW AND ADJUSTED $P$-VALUES. SIGNIFICANCE LEVEL IS $0.05$. THE $P$-VALUES WERE ADJUSTED WITH THE BENJAMINI HOCHBERG PROCEDURE METHOD IN ORDER TO CONTROL THE FDR ERROR.

Let us first consider *gen-ind*. Since this sequence correspond to the independence model there should be no significant episodes. However, since we are accepting $5\%$ of false significant episodes we should expect about $240$ significant episodes. The number of significant episodes discovered is about $10\%$. The higher number for these tests can be explained by the fact that the model we are using is actually trained from the training data and hence contain some error. Should we use the exact model, then the number of significant episodes will drop to $5\%$. After adjusting the raw $P$-values, no significant episode remained. Hence, our method did not find any significant episode from *gen-ind*, as expected.

Next we will consider the sequence *gen-co*. Here we expect to find significant patterns, since the sequence does not obey the independence model. We see from Table III that this is the case. Even after the adjustment there is a considerate amount of significant episodes. By studying the results we found out that the significant episodes had either the basic form of $i \rightarrow i+5$ or a combination of these. This is an expected result since the sequence had $i$ and $i+5$ abnormally close to each other. An important observation here is that the algorithm also discovers complex episodes to be important. Namely, the $P$-value, our quality measure is fair for simple and more complex episodes.

Our next sequence was *moby*. Since the alphabet in this sequence is quite large, the number of candidate episodes is rather small and a lot of these candidate episodes are in fact singletons — in the end $137$ episodes were given a $P$-value. Out of these episodes about $15\%$ were significant based on raw $P$-value and about $10\%$ when $P$-values were adjusted. Some examples among the most significant episodes based on one-sided test were $(white \rightarrow whale)$, $(sperm \rightarrow whale)$, $(old \rightarrow man)$, along with their parallel versions. Such episodes imply that these words occur abnormally close to each other. On the other hand, candidate episodes such as $(time, whale)$ or $(ship, man)$ were not considered significant. This means that even though these combinations occur often, the episode can be explained by the fact that their individual words are common. Similarly, some of the significant episodes discovered in *abstract* were $(research \rightarrow project)$ and $(undergraduate, student)$. Episodes discovered from *address* were $(united \rightarrow states)$, $(united, states)$ and $(fellow, citizen)$.

## VIII. RELATED WORK

Our approach resembles the approach taken in [7], [8] in which the authors considered episode to be significant if the episode occurs too often or not often enough in a fixed window. As a background model the authors used independence model in [7] and markov-chain model in [8]. The main difference between our approach and theirs is that we are studying the behavior of the minimal windows. As we have discussed in the introduction we believe that using the statistics based on minimal windows has an advantage over the fixed window approach.

In [9], the author proposed a criterion for episodes by requiring that the consecutive symbols in a sequence should only within a specified bound. While this approach attacks the problem of fixed windows, it is still a frequency-based measure. This measure, however, is not antimonotonic as it is pointed out in [10]. It would be useful to see whether we can compute an expected value of this measure so that we can compute a $P$-value based on some background model.

In a related work [11] the authors considered parallel episodes significant if the smallest window containing each occurrence of a symbol of an episode had a small value. Their approach differ from ours since the smallest window containing a fixed occurrence of a symbol is not necessarily the minimal window. Also, they consider only parallel episodes whereas we consider more general DAG episodes. An interesting approach has been also taken in [12] where the authors define a windowless frequency measure of an itemset within a stream $s$ to be the frequency starting from a certain point. This point is selected so that the frequency is maximal. However, this method is defined for itemsets and

it would be fruitful to see whether this idea can be extended into episodes.

Finite state machines have been used in [13], [14] for discovering episodes. However, their goal is different than ours since the actual machine is built upon a sequence and not the episode set and it is used for discovering episodes and not computing the coverage.

## IX. Discussion and Conclusions

In this paper we proposed a new quality measure for the episodes. Our approach tackles simultaneously problems with fixed windows but also allows us to incorporate background knowledge. The measure itself is a deviation of the average length of the minimal windows when compared to the expected length according to the independence model.

Our main technical contribution is the technique for computing the distribution of lengths of minimal window. In order to do that we create an elaborate finite state machine and compute probabilities iteratively starting from simple episodes and moving toward complex ones. Once the distribution is computed we are able to perform a statistical test on the discovered minimal windows from the test sequence. Our experiments with the text data suggest that this measure finds significant episodes while ignoring uninteresting ones.

The proposed method requires a parameter $K$, a limit to the size of a minimal window. In this paper we simply have assumed that this parameter is domain-specific and is provided by the user. Setting this parameter high may allow us to discover more interesting patterns. However, when using large values for $K$, computing the model may become computationally infeasible as we are forced to use exact rational numbers in order to guarantee numerical stability. This computational problems may be solved by simulating the independence model instead of computing the exact probabilities. In such case, more analysis is needed to determine a proper number of steps in this simulation.

As a future work we also consider more elaborate models such as Markov Chains. This has been done in [8] for windows of fixed size and our goal is to extend this approach for minimal windows.

Our experiments revealed an interesting behavior within certain sets. Certain information tend to repeat in several forms of episodes. For example, we found that both $(white, whale)$ and $(white \rightarrow whale)$ were significant. This suggests that there is a need for pattern reduction techniques. Such techniques are well studied in the setting of itemsets but are not that well developed for episodes.

## References


[1] H. Mannila, H. Toivonen, and A. I. Verkamo, "Discovery of frequent episodes in event sequences," *Data Min. Knowl. Discov.*, vol. 1, no. 3, pp. 259–289, 1997.

[2] P. Billingsey, *Probability and Measure*, 3rd ed. John Wiley & sons, 1995.

[3] A. W. van der Vaart, *Asymptotic Statistics*, ser. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1998.

[4] S. Laxman, P. S. Sastry, and K. P. Unnikrishnan, "A fast algorithm for finding frequent episodes in event streams," in *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM, 2007, pp. 410–419.

[5] G. I. Webb, "Discovering significant patterns," *Machine Learning*, vol. 68, no. 1, pp. 1–33, 2007.

[6] Y. Benjamini and D. Yekutieli, "The control of the false discovery rate in multiple testing under dependency," *Annals of Statistics*, vol. 29, no. 4, pp. 1165–1188, 2001.

[7] R. Gwadera, M. J. Atallah, and W. Szpankowski, "Reliable detection of episodes in event sequences," *Knowl. Inf. Syst.*, vol. 7, no. 4, pp. 415–437, 2005.

[8] ——, "Markov models for identification of significant episodes," in *SDM*, 2005.

[9] G. Casas-Garriga, "Discovering unbounded episodes in sequential data," in *Knowledge Discovery in Databases: PKDD 2003, 7th European Conference on Principles and Practice of Knowledge Discovery in Databases*, 2003, pp. 83–94.

[10] N. Méger and C. Rigotti, "Constraint-based mining of episode rules and optimal window sizes," in *Knowledge Discovery in Databases: PKDD 2004, 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, 2004, pp. 313–324.

[11] B. Cule, B. Goethals, and C. Robardet, "A new constraint for mining sets in sequences," in *Proceedings of the SIAM International Conference on Data Mining, SDM 2009*, 2009, pp. 317–328.

[12] T. Calders, N. Dexters, and B. Goethals, "Mining frequent itemsets in a stream," in *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007)*, 2007, pp. 83–92.

[13] Z. Tronícek, "Episode matching," in *Combinatorial Pattern Matching*, 2001, pp. 143–146.

[14] M. Hirao, S. Inenaga, A. Shinohara, M. Takeda, and S. Arikawa, "A practical algorithm to find the best episode patterns," in *Discovery Science*, 2001, pp. 435–440.