Just Wait For It...
Mining Sequential Patterns with Reliable Prediction Delays

Joscha Cüppers Jilles Vreeken
CISPA Helmholtz Center for Information Security, Saarbrücken Germany
{joscha.cueppers, jv}@cispa.de

***Abstract*—Suppose we are given an event sequence X of observed events and an equally long binary sequence Y that indicates whether something of interest happened at that particular point in time. We consider the problem of mining sequential patterns from X that reliably predict those interesting events. With *reliable* we mean those patterns that not only predict that an interesting event is likely to follow but especially those patterns for which we can with high precision tell how long until that event will happen. That is, we are after patterns that have highly skewed distributions of delays between pattern occurrences and predicted events. In particular, we are after the smallest, least redundant set of such patterns that together explain the interesting events well.**

We formally define this problem in terms of the Minimum Description Length principle, by which we identify the best patterns as those that describe the data most succinctly. As discovering the optimal explanation of Y given a set of patterns, as well as the discovery of optimal pattern set are both hard problems that do not allow for straightforward optimization, we propose the heuristic OMEN algorithm. Through extensive empirical evaluation we show that OMEN works well in practice and beats the state of the art both quantitatively and qualitatively.

I. INTRODUCTION

Suppose we are given an event sequence X of observed events and an equally long binary sequence Y that indicates those points in time where something happened that we want to predict—earthquakes, for example. We consider the problem of mining a small set of interpretable and actionable patterns from X that reliably predict those interesting events. With *reliable* and *actionable* we mean those patterns that not only highly accurately predict *that* an interesting event will follow, but especially those for which we can tell with high precision *how long* it will be until the predicted event will happen. That is, we are after patterns that have a compact distribution of delays between pattern occurrences and predicted events. As real processes are rarely trivial, we consider the problem of discovering a small and non-redundant set of patterns that *together* reliably predict the interesting events.

Event prediction is well-studied in time series analysis. Most work focuses on continuous-valued data, however, and particularly considers detecting abrupt distributional changes [15] and identifying events that precede such changes [27]. As we aim to discover patterns that explain the interesting time points, our work is closer to that of sequence classification [35]; we can interpret the interesting events in Y as labels, by which the task is to find those patterns that predict these. Existing solutions, however, focus purely on discovering all patterns that sufficiently accurately predict that an interesting event will follow *some time* after their occurrence [36], rather than our goal of discovering a small set of patterns for which we can reliably say *how long* it will take before that event occurs. As such, our work is also related to information flow [28] and Granger causality [13], in the sense that patterns are only interesting if their occurrences provide significantly more information about Y than the history of Y does by itself.

We formalize the problem in terms of the Minimum Description Length (MDL) principle [25], by which we identify the best patterns in X as those that describe Y most succinctly. We model the data such that for every pattern occurrence we have to encode the delay until the associated interesting event. The more peaked this distribution, the cheaper it will be to encode the delay, and hence we particularly favor patterns that accurately predict both the occurrence as well as the time until an interesting event. Discovering the optimal explanation of Y given a set of patterns, as well as the discovery of optimal pattern set, are both hard problems, and neither allow for straightforward optimization. To efficiently discover good pattern sets in practice, we propose OMEN, a greedy heuristic that iteratively optimizes the alignment of pattern occurrences to interesting events, and uses this alignment to discover the best refinements of the patterns. OMEN does not have any hyper parameters, does not impose restrictions on the delay distribution, and does allow for overlap between predictions.

We evaluate OMEN in practice on both synthetic and real-world data. We show that the OMEN score reliably determines the predictiveness of patterns, and compares favorably two state-of-the-art information flow scores [28], [5]. We show that the OMEN pattern miner reconstructs the ground truth both in terms of predictive patterns as well as their delay distributions, outperforming four supervised and unsupervised sequential pattern miners [30], [10], [31], [35]. On real-world data we confirm that OMEN discovers meaningful patterns.

This paper is structured as usual. We start with notation and preliminaries in Sec. II, after which we formally introduce the problem in Sec. III. We present the OMEN algorithm in Sec. IV and discuss related work in Sec. V. We empirically evaluate in Sec. VI, and round up with discussion and conclusions in Secs. VII and VIII. We make all code and data publicly available,[1].

[1] http://eda.mmci.uni-saarland.de/prj/omen/

II. Preliminaries

We start by introducing the notation and discussing preliminaries we will use throughout the paper.

A. Notation

We consider finite-length discrete event sequences X of length n over a finite alphabet $\Omega = \{a, b, \ldots\}$, i.e. $X \in \Omega^n$. We write $|X| = n$ to denote the length of the sequence, $X[i]$ to refer to the i^{th} event in X, and $X[i:j]$ to denote the subsequence $X[i], ..., X[j]$. We write $||X||_a$ for the number of times we see event $a \in \Omega$ in X.

As data, we consider two equally long event sequences X and Y, where X encodes the observed events over Ω_X, and Y is a binary sequence, $\Omega_Y = \{0, 1\}$, in which a 1 encodes that something 'interesting' happened at that point in time.

We consider sequential patterns $s \in \Omega^m$, where $m = |s|$ is the length of the pattern. We say a pattern s occurs in, or matches $X[j]$ iff $X[j-m:j] = s$. Given an event sequence X and pattern s we can construct a binary sequence Z_s where all $Z_s[j] = 1$ iff s matches $X[j]$. A *predictive* pattern s is a sequential pattern s with an associated discrete delay distribution P_s that specifies the probability density $P_s(\delta)$ that something interesting will happen δ time steps after an occurrence of the pattern.

All logarithms are base 2, and we use the common convention that $0 \log 0 = 0$.

B. Minimum Description Length

The Minimum Description Length (MDL) principle [14] is a computable and statistically well-founded model selection criterium based on Kolmogorov Complexity [20]. For a given model class $\mathcal{M}$, it identifies the best model $M \in \mathcal{M}$ as the one that minimizes the number of bits needed to describe both model and data, $L(M) + L(D|M)$ where $L(M)$ is the length of model M in bits and $L(D|M)$ the length of data D given M. This is known as two-part, or crude MDL—in contrast to one-part, or refined MDL [14], which is not computable for arbitrary models. More specifically, we use two-part MDL because we are particularly interested in the model: those patterns that predict. In MDL we are never concerned with materialized codes and only care about code lengths. To use MDL we have to define a model class $\mathcal{M}$, and encodings for data and model. Below we present our encoding and model definition for the problem of discovering predictive patterns.

III. Theory

In this section, we introduce the problem at hand. Before stating the problem formally, we will define a model class, and show how to encode a model and data given a model.

A. The Problem, Informally

We are interested in discovering that set of predictive sequential patterns s that most reliably predict the interesting events in Y given observed events X. Our models consist of sequential patterns s and associated delay distributions P, i.e. $M = \{(s_1, P_{s_1}), (s_2, P_{s_2}), \ldots, (s_k, P_{s_k})\}$. Given s and X we have a binary sequence Z_s to mark those time points where s occurs in X and every such occurrence predicts (in principle) that an interesting event is about to happen in Y. We call the mapping of occurrences of a pattern s to the actual interesting events it predicts in Y its alignment A_s. Formally an alignment A_s is a sequence of time delays of length $||Z_s||_1$.

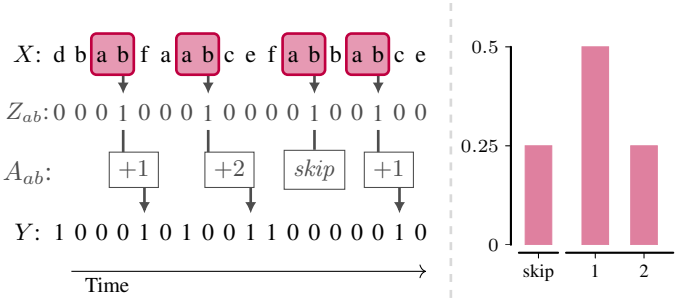

Figure 1: Encoding example (left) and time delay distribution P_{ab} (right) of pattern ab. Toy example where pattern ab covers 3 out of 5 interesting events in Y.

We allow for noise in both X as well as in Y, and hence not all predictions are required to be successful; it is allowed that a prediction is 'skipped'. We interpret a 'skip' as an infinite delay. A delay distribution provides the probabilities of an interesting event occurring δ time steps after an occurrence of pattern s. The higher the probability of δ, the fewer bits we will need to encode that value. Overall, the less we have to 'skip' and the more peaked the distribution is, the more cost effective, in terms of bits per interesting event, we can describe A_s, and hence the more succinctly we will be able to describe Y.

As an illustrative example, consider the toy setting depicted in Fig. 1. We show an event sequence X of length 18 over an alphabet $\Omega = a, \ldots, f$ in which there are four occurrences of pattern ab. We show both the occurrence sequence Z_{ab}, as well as an example alignment of these occurrences to the interesting events in Y. We show the corresponding delay distribution on the right, which shows that in 50% of the cases an interesting event happens one time step after the occurrence of the pattern, as well as that it has a 25% probability of false predicting the occurrence of an interesting event (skip).

We allow Y to be complex, in the sense that multiple patterns may be needed to reliably predict all interesting events. In other words, we have to allow every pattern $s \in M$ to predict only some of the interesting events in Y. We write $\hat{Y}_s$ for the binary sequence that indicates which interesting events in Y that s predicts. Formally, $\hat{Y}_s = A_s \circ Z_s$, where $\circ$ shifts the i^{th} 1 in Z_s, to the right, by $A_s[i]$ positions.

Ideally, the patterns in M together predict all interesting events in Y, $\hat{Y} = \bigvee_{s \in M} \hat{Y}_s = Y$. However, due to noise not all interesting events may have good explanations, and further, not every model will be able to explain all interesting events, and hence in addition to M and A, we will need to transmit the residual sequence R that is defined as the bit-wise XOR between the predicted $\hat{Y}$ and the true Y, $R = \hat{Y} \oplus Y$.

B. MDL for Predictive Sequential Patterns

Based on the intuition above, we will now formalize our score. We will first discuss how to encode the interesting event sequence Y given a model M and observed events X, and afterwards detail how to encode a model M.

1) Encoding the Data given a Model: Given X and the patterns $s \in M$, it is straightforward to construct pattern occurrences sequences Z_s. To determine the $\hat{Y}_s$ from Z_s we need to know the delays between pattern occurrences and predicted events. That is, we need the alignments A_s.

To encode an alignment A_s for pattern s, we simply have to transmit the delay δ for every occurrence of s in X. We do so using optimal prefix codes [8] over the time delay distribution P_s, by which we have

$$L(A_s \mid P_s) = -\sum_{i=0}^{|A_s|} \log P_s(A_s[i]) \ .$$

Once we know the alignments, we can reconstruct the $\hat{Y}_s$ and therewith $\hat{Y}$. To reconstruct Y without loss, we need the residual sequence R. We have

$$L(R) = L_{\mathbb{N}}(||R||_1) + \log \binom{|X| \ - ||\hat{Y}||_1}{||R||_1} \ ,$$

where we first encode the number of 1s in R using $L_{\mathbb{N}}$, the MDL-optimal encoding for integers [26]. It is defined as

$$L_{\mathbb{N}}(z) = \log^* z + \log c_0$$

where $\log^* z$ is defined as $\log z + \log\log z + \ldots$, only including the positive terms in the sum. To obtain a valid encoding, i.e. satisfy the Kraft inequality, we set $c_0 = 2.865064$ [26]. Now that we know the number of 1s in R, we can optimally encode the actual sequence R via an index over a canonically ordered set of all sequences of length n with $||R||_1$ 1s. Since we already know the location of predicted interesting events, we don't have to consider these in the residual, therefore we can remove $||\hat{Y}||_1$ possible locations. As the binomial coefficient greatly increases with every additional interesting event in R we favor residuals with fewer interesting events.

Putting these two parts together, we have

$$L(Y \mid M, X) = \left(\sum_{(s,P_s) \in M} L(A_s \mid P_s) \right) + L(R)$$

for the number of bits to encode Y given a model M and observed events X.

2) Encoding a Model: Next, we formalize how we encode a model $M \in \mathcal{M}$ in bits. At a high level we have

$$L(M) = L_{\mathbb{N}}(|M|) + \sum_{(s,P_s) \in M} L(s) + L(P_s) \ ,$$

where we first encode the number of patterns in the model, and then the patterns and their associated delay distributions.

Patterns are essentially just a sequence of k events from Ω. We use $L_{\mathbb{N}}$ to encode their length, and to avoid any bias towards events $e \in \Omega$, we encode the actual events in s using an index over Ω. Thus the cost of one pattern is

$$L(s) = L_{\mathbb{N}}(|s|) + |s| \log |\Omega| \ .$$

To encode a time delay distribution, it suffices to encode which time deltas have a probability greater zero, and then encoding how likely each of these deltas are. We write $\Delta_s = \{\delta \mid P_s(\delta) > 0\}$ for the set of δ values with non-zero probability. Formally, we then have

$$L(P_s|X) = L_{\mathbb{N}}(\delta^*) + \log(\delta^*) + \log \binom{\delta^* + 1}{k} + \log \binom{||Z_s||_1 - 1}{k - 1} \ ,$$

where we first encode interval of possible deltas, $[0, \delta^*]$, by encoding the value of the largest delta with non-zero probability, $\delta^* = \max(\Delta)$, using $L_{\mathbb{N}}$. As we are not interested in 'instantaneous' predictions, we repurpose $\delta = 0$ as 'skip'. Next we encode the number of δs with non-zero probability mass, $k = |\Delta|$, for which we need $\log \delta^*$ bits. We then encode the values $\delta \in \Delta$ by an index. The more gaps, $\delta' s$ where $P_s(\delta) = 0$, the higher this cost.

After which we only have to specify the probability mass per δ. This reduces to an index over a number composition, i.e. an index over every possible way to distribute the $||Z_s||_1$ occurrences (balls) over k non-empty bins. The more deltas we have to consider the higher the cost will be, hence we prefer distributions with fewer deltas.

This concludes the description of a lossless MDL encoding for a model M.

C. The Problem, Formally

With the above we can now formally state the problem.

The Minimal Prediction Problem *Given event sequence X over alphabet Ω and binary sequence Y indicating time points of interest, find that set of predictive sequential patterns and associated time delay distributions $M = \{(s_1, P_{s_1}), (s_2, P_{s_2}), \ldots, (s_k, P_{s_k})\}$ and that alignment A of pattern occurrences to interesting events in Y, such that the total encoded length*

$$L(Y, M \mid X) = L(M) + L(Y \mid M, X)$$

is minimal.

To solve this problem exactly we have to consider a rather large search space; as we do not wish to limit the length of a pattern beforehand, patterns can be up $|X| - 1$ long, resulting in $\sum_{i=1}^{|X|-1} |\Omega|^i$ possible patterns. Per pattern s there exist $(||Y||_1 + 1)^{||Z_s||_1}$ possible alignments. This leaves the final part selecting a set of of pattern, alignment tuples. We can limit the number of tuples in our model to $||Y||_1$. Combined this gives us

$$\sum_{j=0}^{||Y||_1} \binom{\sum_{s \in S} (||Y||_1 + 1)^{||Z_s||}}{j}$$

Algorithm 1: FINDALIGNMENT

input : binary sequence Z_s and binary sequence Y
output: alignment A_s between Z_s and Y

1 $A_s \leftarrow$ align each i where $Z_s[i] = 1$ to the next j where $Y[j] = 1$
2 $A_s \leftarrow$ for each j where $Y[j] = 1$, only keep alignment with highest $P_s(j - i)$
3 $A_s \leftarrow$ for each $Z_s[i] = 1$ aligned to a skip, align to $Y[j] = 1$ that $\max P_s(j - i)$
4 **while** $L_s(Y)$ decreases **do**
5 for each $(i, j) \in A_s$ where $P_s(j - i) = \min P_s(\delta)$, align i to $Y[j] = 1$ that $\max P_s(j - i)$
6 **return** A_s

possible solutions, where S is the set of all patterns. Unfortunately, this search space does not exhibit structure such as (weak) monotonicity, convexity, or submodularity, that we could exploit to guide our search.

Hence, we resort to heuristics.

IV. ALGORITHM

In this section we present the OMEN algorithm to heuristically solve the Minimal Prediction Problem. We approach this by splitting the problem into two parts. First, given a pattern we aim to find its delay distribution by optimizing the alignment between pattern occurrences and interesting events. Second, we consider the problem of discovering good patterns. We discuss these steps in turn.

A. Discovering Alignments and Delay Distributions

We start by discussing how to optimize a pattern delay distribution P_s — or equivalently, an alignment A_s — for a given pattern s. That is, how to minimize $L(Y, \{(s, P_s)\} \mid X)$. To enhance readability, wherever clear from context, we will write $L_s(Y)$ for $L(Y, \{(s, P_s)\} \mid X)$, and $L(Y)$ for the length of Y under the null model. As we have seen above, solving $L_s(Y)$ exactly requires testing all possible alignments, and is hence computationally unfeasible. We will therefore heuristically minimize $L_s(Y)$, for which in Algorithm 1, we present a simple yet effective approach.

To allow a simpler representation we overload the notation of alignment A_s and represent the alignment as a set of tuples (i, j), where i is the location of the pattern match in X and j the location of the aligned interesting event (or 'skip'). This can easily be transformed into a sequence of deltas by ordering the elements by i and subtracting i from j.

FINDALIGNMENT consists of four main steps, that each result in a new alignment. First, based on the assumption that each pattern occurrence predicts the directly following interesting event, we simply align every occurrence $X[i] = s$ of pattern s in X to that interesting event in Y that is closest in time but at least one time step into the future, i.e. $A_s = \{(i, j) \mid X[i] = s, \arg\min_{j>i} Y[j] = 1\}$ (line 1), and determine the resulting delay distribution from the alignment. Some interesting events may now be aligned with multiple pattern occurrences, which inherently is not a problem but redundant nevertheless. It also 'blocks' these pattern occurrences from predicting other interesting events. We hence re-align all such pattern occurrences to 'skip', except for the one with maximal $P_s(j - i)$, and re-infer the delay distribution (l. 2). Next, we use the new distribution to align every pattern occurrence mapped to 'skip' to that interesting event, $Y[j] = 1$, which maximizes $P_s(j - i)$. If there is no interesting event with non-zero probability under P_s, we map it to 'skip' (l. 3).

This gives us an initial alignment that we can optimize towards $L_s(Y)$. We do this by reassigning all pattern occurrences mapped to interesting events with minimal $P_s(\delta)$, to interesting events with maximal $P_s(\delta)$. If there is no interesting event where $P_s(\delta) > \min P_s(\delta)$, we map it to 'skip' (l. 5). We repeat this step until $L_s(Y)$ no longer decreases. We can now, given a pattern, find an alignment between Z_s and Y. For the remaining section $L_s(Y)$ denotes the length of Y under pattern s with the alignment given by the above described method.

B. Discovering Good Sets of Patterns

With the above we know how to find an alignment for a given pattern, and hence how to score a pattern. We will now discuss how to find good patterns. As is usual, we consider a greedy bottom-up approach. We immediately face a problem, however. If a pattern s does not encode Y in fewer bits than under the null model, the best alignment is given by mapping all pattern occurrences to 'skip', which amounts to the null model of encoding all of Y in the residual. That is, we cannot use our score to identify whether a pattern is 'promising' unless it is already 'good enough', and as only in trivial cases singleton events in X will help to compress Y we cannot start by adding 'good' singletons and then refine them.

Rather than resorting to exhaustively scoring every possible pattern s under some arbitrary constraints, we define an optimistic estimator $\bar{L}_s(Y)$ by which we estimate the $L_{s'}(Y)$ of the theoretically best possible refinement s' of s which has exactly that subset of occurrences of s that align best with Y. To find out which set of pattern occurrences are the 'right' instances, i.e. the best compressing ones, would however, require testing all possible combinations and hence result in an unfeasible runtime. We therefore instead estimate via aligning Z_s and Y as described above, and treat pattern occurrences mapped to 'skip' as if they do not exist, in our encoding this is equivalent to setting the encoding cost of 'skip' to zero. Hence, $\bar{L}_s(Y)$ gives the length of Y where we set $P_s(skip) = 1$ for the encoding of A_s and $P_s(skip) = 0$ for the encoding of P_s, as if only the pattern occurrences aligned to interesting events exist.

With $\bar{L}_s(Y)$ we can now estimate whether it is possible to extend a (possibly currently non-compressing) pattern s to a pattern s' that would improve our model. We give the pseudo code of the OMEN pattern miner as Algorithm 2. We start with an empty model where all interesting events are unexplained by patterns, i.e. encoded via residual R (line 1). The main idea

Algorithm 2: OMEN

input : event sequence X and binary sequence Y
output: model M, set of pattern delay distribution pairs

1 $M \leftarrow \emptyset$; $R \leftarrow Y$
2 $C \leftarrow \Omega$; $C' \leftarrow \emptyset$; $S \leftarrow \emptyset$
3 **while** C is not empty **do**
4 **foreach** $s \in C$ **do**
5 **if** $\bar{L}_s(R) < L(R)$ **then**
6 $C' \leftarrow C' \cup \{s + e, e + s \mid \forall e \in \Omega\}$
7 **if** $L_s(R) < L(R)$ **then**
8 **add** s to S
9 **foreach** $s \in S$ ordered by $L_s(R)$ **do**
10 **if** $L_s(R) < L(R)$ **then**
11 $s', P_{s'} \leftarrow$ REFINE PATTERN(s)
12 **add** $(s', P_{s'})$ to M
13 $\hat{Y} \leftarrow$ compute from M
14 $R \leftarrow \hat{Y} \oplus Y$
15 $C \leftarrow C'$; $C' \leftarrow \emptyset$; $S \leftarrow \emptyset$
16 **return** M

Algorithm 3: REFINE PATTERN

input : predictive pattern s,
output: greedy refinement of pattern s and delay distribution P_s

1 $s* \leftarrow s$ $s' \leftarrow s$
2 **while** $\bar{L}_s(R) < L_{s'}(R)$ **do**
3 $H \leftarrow \{(i,j) \in A'_s \mid j \neq skip, X[i+1] = s'e\}$
4 $T \leftarrow \{(i,j) \in A'_s \mid j \neq skip, X[i] = es'\}$
5 **if** $\max_{e\in\Omega} |T| > \max_{e\in\Omega} |H|$ **then**
6 $s' \leftarrow s' + \arg\max_{e\in\Omega} |T|$
7 **else**
8 $s' \leftarrow \arg\max_{e\in\Omega} |H| + s'$
9 **if** $L_{s^*}(R) > L_{s'}(R)$ **then**
10 $s^* \leftarrow s'$
11 **return** (s^*, P_{s^*})

is to then iteratively add patterns to the model M that predict some of the events in R, by which we focus the search on those interesting events that are not yet predicted.

Starting from the singletons as candidate set C, we take a breadth-first search approach where we extend all candidates that are promising with regard to our optimistic estimate (line 4–6) and identify those that help to better compress Y (l. 7–8). As extensions of promising patterns we consider all patterns s' where we add any single symbol from alphabet Ω at either the end or beginning of s (l. 6).

Next, we iteratively consider adding the candidates $s \in S$ that passed the compression-check into the model (l. 9–14). We do so in order of how much they help to compress (l. 9) and to avoid redundancy only consider those that indeed improve the score (l. 10) — for example, if $abcd$ is the true pattern and $ab \in S$ and $cd \in S$ then both (probably) explain the same interesting events. Once we added one, the other does not offer any additional information. When adding a pattern to our model we first refine it in a depth-first manner (l. 11) which we discuss below. After adding a pattern we recompute the residual (l. 13–14). We repeat these steps until we have no further candidates (l. 3), and then return the final model M.

The refinement of patterns goes hand in hand with our estimator. That is, for those patterns that already give a gain in compression, we search for their best possible refinement. Rather than exhaustively exploring every possible super-pattern of a candidate s, the key idea is to greedily extend the pattern until our optimistic estimator no longer estimates a better possible pattern (or, we run out of possible extensions).

We give the pseudo-code as Algorithm 3. We start with a pattern s (l. 1) and consider the most frequent out of the refinements es and se (l. 5). For both we choose event $e \in \Omega$ that maximizes frequency. That is, rather than merely maximizing frequency, we only count events e that are adjacent to pattern occurrences that are currently aligned to an interesting event (l. 3–4). The key idea is that extending a pattern makes it more specific, and hence reduce recall—while, by maintaining the current predictions, we maximize precision.

We repeat this process until our optimistic estimator no longer gives a better estimation than the best seen pattern up to this point. We then return that pattern with lowest $L_s(R)$. This concludes the description of OMEN.

C. Complexity

Next we consider the complexity of OMEN. In the worst case OMEN considers every possible sequential pattern over Ω, which means a complexity of $\mathcal{O}(|\Omega|^{|X|-1})$. For every pattern we have to find an alignment. The alignment algorithm and the subsequent optimization is dominated by picking the most likely δ for each pattern occurrences aligned to a 'skip'. In the worst case we have to consider all possible $\delta \in P_s$. This gives a complexity $\mathcal{O}((||Z_s||_1)^2)$, as in the worst case our distribution has a different time delta for every pattern occurrence, i.e. $||Z_s||_1$ time deltas.

When adding a pattern to our selection we refine it to the best version we can find. We do this by greedily, but in the worst case we have to consider up to $|X|-1$ patterns. For each considered pattern we have to go over X to count the adjacent symbols, combined this gives us a complexity of $\mathcal{O}(|X|^2)$.

Although the overall complexity is therewith quite horrible in the worst-case, as real data is sparse and our MDL score protects against overfitting, OMEN is fast in practice, taking seconds up to minutes in our experiments.

V. RELATED WORK

Our work is related both to prediction and information flow in time series, as well as to pattern mining.

At its core, the OMEN score aims to measure how the occurrences of a pattern s in X help to reliably predict the occurrences of interesting events in Y. As such, it is strongly related to Granger causality [13]. Granger causality is based on the idea that if the past of a time series Z does help to predict Y, given the past of Y, it "Granger" causes Y. Linear [13] and nonlinear [6] scores have been proposed, whereas others studied the effects of events on time series [27]. Transfer Entropy (TENT) [28] and CUTE [5], are both information-theoretic instantiations of Granger causality for discrete data, where the one measures information flow in terms of entropy, and the other in terms of MDL. In the experiments we will compare the OMEN score to both.

Prediction in time series [19], [36], [33], [7], [34] is a classic research topic to which OMEN is related. A prototypical example is failure prediction, where the goal is to predict upcoming failures with sufficient time to act on the prediction. By far most work focuses on the case where X is continuous valued and the goal is to discover time points where the distribution of X changes [32], [15]. Another popular task is the prediction of upcoming events based on social media activities and text [24], [12].

Discovering interpretable sequential patterns has been extensively studied [1]. We can differentiate between two settings, based on the notion of support of a pattern. The first, where we have a database of sequences and support is defined by the number of instances containing a pattern [31], and the second where we have one or multiple sequences where support is defined as the number of occurrences within a sequence, measured using either a sliding window [22] or counting the number of minimal windows [18]. Both settings have been studied in detail, especially for the goal of mining all (closed) frequent patterns [31].

Recently, research focused more on discovering small, non-redundant sets of patterns that generalize the data, for which MDL based approaches, such as SQS [30], SQUISH [4], and DITTO [3] have been shown to be effective. Instead of taking a descriptive approach, Fowkes and Sutton [10] proposed a method to discover small sets of informative patterns based on a generative model. One step closer to our goal, Galbrun et al. [11] recently studied the problem of discovering sequential patterns that periodically appear. Although all related to OMEN in the sense that they also discover small sets of patterns, these methods are all strictly unsupervised. We will compare to both SQS [30] and ISM [10].

Identifying patterns that predict the occurrence of events can be approached as a supervised sequence classification problem, where given a labeled sequence database the goal is to find those sequential patterns that allow a classification [2], [35], [9]. SCIS [35], a recently proposed rule-based sequence classifier, to this end filters the top-k best classifying rules out of many candidates. We consider a different setting, where instead of a sequence database, we only have two sequences X and Y, where Y can be interpreted as our labels. We include a comparison to SCIS in the experiments.

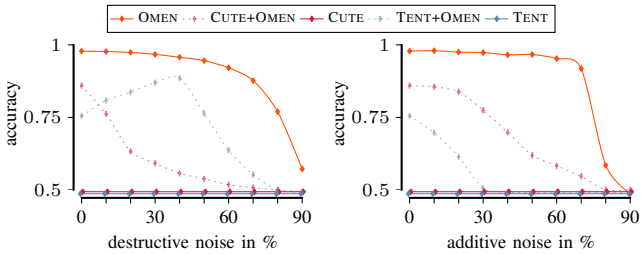

Figure 2: [Higher is Better] OMEN reliably determines predictiveness of patterns, both for destructive (left) and additive (right) noise. Dashed lines are the results of TENT and CUTE on the predicted $\hat{Y}_s$ for Z_s as discovered by OMEN.

VI. EXPERIMENTS

In this section, we will empirically evaluate OMEN on synthetic as well as real-world data. We compare OMEN to TENT [28] and CUTE [5] to determine how well it can tell predictive from non-predictive patterns, as well as to SQS [30], BIDE+ [31], ISM [10], and SCIS [35] to determine how well it can discover predictive patterns from data. We implemented OMEN in Python and provide the source code for research purposes, along with the used datasets and experiment specifications with generator.[2] All experiments were executed single-threaded on machines with two Intel Xeon CPU E5-2643 processors and 256 GB of memory, running Linux. We report wall-clock running times.

A. Synthetic Data

To assess performance against known ground truth we consider synthetic data, which we generate as follows. We first generate event sequence X of length $n = 500\,000$ by uniformly at random drawing n events from an alphabet Ω of length 26, i.e. $X \in \Omega^n$, and initialize $Y = \{0\}^n$. We add structure by planting 50 predictive and 25 non-predictive patterns. Every pattern is generated independently, where we first draw its length l from $[2, 6]$, its events $s = \{e_1, \ldots, e_l\}$ from Ω, and its frequency f from $[25, 1000]$, again all uniformly at random. We plant patterns into X by sampling u.a.r. f insertion positions $i \in [0, n]$, where we simply overwrite the existing values in X, i.e. $X[i : i + l] = s$. To ensure the ground truth holds, we do not overwrite previously planted patterns and remove any accidental occurrences of patterns in X by resampling those events. For the predictive patterns s we additionally generate interesting events in Y by, per insertion position i, sampling a delay δ u.a.r. from $[10, 15]$ and setting $Y[i + l + \delta] = 1$. Finally, we add noise to the data by flipping values in Y. We consider both destructive noise, where we flip 1s to 0s, as well as additive noise, where we flip 0s to 1. Unless specified otherwise, all results on synthetic data are averaged over 20 independently generated datasets.

B. Evaluating the Score

We first evaluate how well OMEN can tell predictive from non-predictive patterns. For OMEN, we consider a pattern to be predictive if it helps to compress Y. We compare OMEN to TENT [28] and CUTE [5], two state of the art methods based on Granger causality that measure how much information Z_s provides towards Y. For both we say Z_s predicts Y if they conclude that Z_s Granger-causes Y. We optimize the lag-parameter of TENT over $[1, 15]$ per experiment.

We generate data with varying amounts and type of noise as described above, and for every planted pattern in every dataset test whether the score considers it predictive or not. We give the average weighted accuracies in Fig. 2. We see that OMEN is able to identify predictive patterns with high accuracies even for large amounts of noise, whereas TENT and CUTE applied on Z_s and Y reduce to a coin flip as they expect all (most) interesting events to be explained by Z_s. When we apply TENT and CUTE not on the raw data Y but rather on the $\hat{Y}_s$ that OMEN discovers as the best explanation of Y given s, we see that their accuracies increase up to 80%, yet remain much worse than those of OMEN.

C. Evaluating the Patterns

Next, we evaluate how well OMEN discovers predictive patterns from data. We compare OMEN to BIDE+ [31], SCIS [35], SQS [30], and ISM [10]. Although BIDE+, SQS, and ISM are unsupervised by nature, we can use them to discover predictive patterns by first splitting X according to the interesting events in Y—so creating a database of sequences leading up to the next interesting event—and subsequently using the OMEN alignment and score to determine which patterns are predictive. As SQS and ISM discover reasonable numbers of patterns we evaluate all. As BIDE+ here discovers 2.9M patterns on average, we only consider the top-200 results. SCIS discovers predictive patterns by itself but requires a labeled sequence database as input. As positive samples we consider the window of w events in X that lead up to each interesting event in Y. As negative samples we then split the remaining data into non-overlapping windows of length w, which avoids skew (in positive and negative samples) as well as bias (non-intersecting with positive samples). We ensure SCIS can discover all planted patterns by setting $w = 20$.

As metric to evaluate success we consider the edit distances between planted patterns q to the best matching discovered pattern s, as this allows us to reward recovery not only of exact matches but also of fragments of ground truth patterns. As an example, if we plant pattern $abcd$ but discover abc as best match, we should not treat it as random. Formally,

$$u_r(T, S) = \sum_{q \in T} \max_{s \in S} 1 - \frac{d(q, s)}{|q|}$$

is our quality metric on which we evaluate recall, with T is the set of planted patterns, S is the set of discovered patterns, and $d(q, s)$ is the standard Levenshtein edit distance between two patterns q and s. We sum only the non-negative terms. To evaluate precision we compute for all $s \in S$ the best match and take the sum over the max $|T|$ elements. Formally, we use

$$u_p(T, S) = \sum_{s \in S^*} s$$

where S^* is the set of the $\max |T|$ elements of $\{\max_{q \in T} 1 - {}^{d(q,s)}/_{|q|} \mid \forall s \in S\}$.

We run all methods on synthetic data generated as above. ISM reports only singletons, and BIDE+ does not return any predictive patterns. We therefore omit them from the analysis. We report the F1 scores of the remaining methods as Figure 3, and provide the precision and recall plots in Figure 7 and Figure 8 in the appendix.

We use the usual definitions for recall and precision where the true positive count is replaced by the respective $u(T, S)$. We see that OMEN outperforms SQS and SCIS by a wide margin, and is especially robust against destructive noise. For additive resp. combined noise, it performs well up to 40% noise. While SQS is a good second, SCIS returns overly many patterns and hence has very low precision.

D. Evaluating the Model

As the final experiment on synthetic data, we evaluate how well OMEN can describe the data at hand, in particular, how close the models that it discovers get to the ground truth. We start with a sanity check, considering data where Y is independent from X. We do so by generating data without noise and then destroying any dependence between X and Y by randomly permuting Y. When we run OMEN on this data it correctly does not report any patterns.

Next, using the same data generating scheme as above, and consider the number of bits that OMEN requires to describe the data, compared to the ground truth model, and the null or 'empty' model where X and Y are described without any patterns. Per noise level we report the worst-case performance, where the difference between discovered and true model is greatest. We give the results as Fig. 4. Overall we see that OMEN returns high quality models and is highly robust against noise. In particular, we see that destructive noise has barely any effect on OMEN's ability to discover good models: in terms of bits, its results are always very close to the ground truth and far from the null model. For additive noise—where increasing numbers of interesting events cannot be explained by patterns in the data—of around 40% we see that OMEN reduces to the null model, which from 70% noise is indeed the most succinct description of the data. For combined noise we see that OMEN discovers models close to the ground truth, up till about 40% noise, after which it correctly returns the null model as the best (simplest) description of the data. These results explain the drop in F1 we observed in the previous experiment, as beyond these noise levels the data simply does not exhibit any significant structure anymore.

To evaluate how well OMEN approximates the ground truth delay distributions, we measure the Jensen-Shannon divergence [8] between the discovered P_s and true P_s^* for

[2]http://eda.mmci.uni-saarland.de/prj/omen/

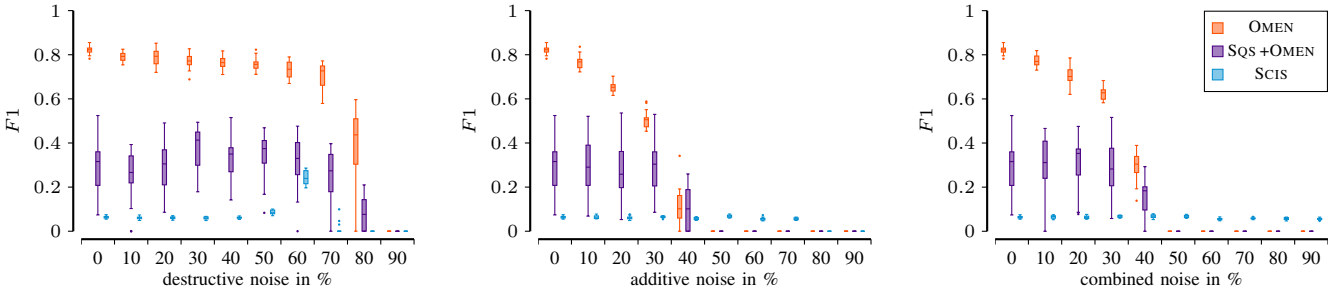

Figure 3: [Higher is better] $F1$ score results on synthetic data for destructive (left), additive (middle) and combined (right) noise. OMEN clearly outperforms SQS and SCIS. Figure 4 shows that the data does not exhibit any significant structure beyond 80% for destructive, 60% for additive and 40% for combined noise. Hence the drop in $F1$ for additive and combined noise.

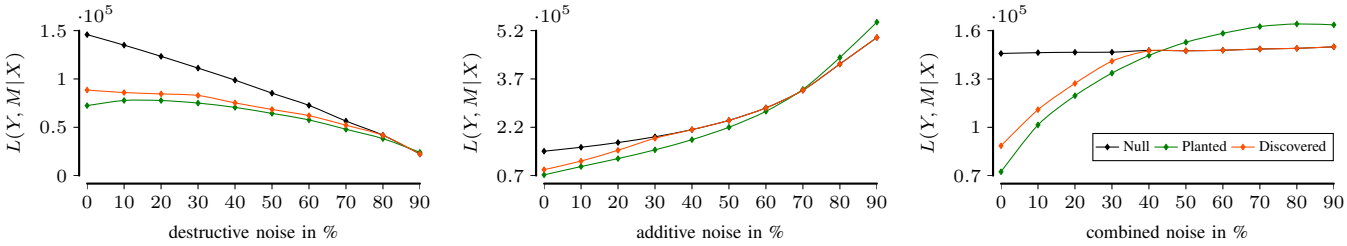

Figure 4: [Lower is better] OMEN discovers models close to the ground truth for destructive (left), additive (middle) and combined (right) noise. Plots show bits needed to encode Y given the null, planted and discovered model. We show for each noise level the experiment with the worst performance, i.e. where the difference, in bits, between discovered and planted model is greatest.

all exactly recovered patterns, averaging over all noise levels. We find that with a worst case divergence of 0.53 bits and an average divergence of only 0.014 bits, OMEN recovers the delay distributions nearly perfectly.

Last, we consider the robustness of OMEN against patterns with noisy delay distributions. To this end we generate synthetic data as above, but, to keep the results interpretable, we now plant only a single pattern of length 6 that predicts 2 000 interesting events. As time delay distribution we consider a Normal distribution with mean 50, and vary the standard deviation from 2 to 60 in steps of 2. We record the number of bits $L(Y, M \mid X)$ needed by resp. the null model, the ground truth model, and the model discovered by OMEN. We give the worst case results per standard deviation, out of 20 experiments, as Fig. 5. We see that OMEN is robust to patterns with wide delay distributions, discovering models that compress better than the null—and hence, discovering the true pattern—up to a delay distribution with a standard deviation of 44. From a standard deviation of 58 onward the null model beats the planted model.

E. Evaluating on Real Data

Last, we evaluate OMEN on real world data. We consider three datasets, electrocardiograms (*ECG*), a daily activities log (*Lifelog*) and water levels combined with precipitation records (*Saar*). We give the basic statistics in Table I. We compare to SQS and SCIS. As SQS allows for gaps, and real world patterns might show these, we interpret its results as minimal windows

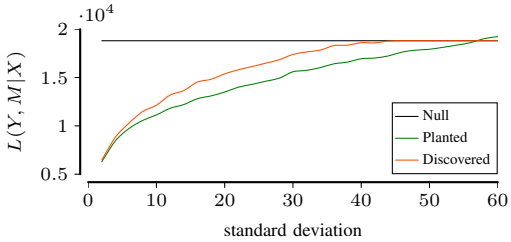

Figure 5: [Lower is better] OMEN recovers a model close to ground truth for a Gaussian time delay distribution. Ground truth model consists of one pattern with gaussian time delay distribution with a median at 50.

for which we again use the OMEN score and alignment to determine which ones are predictive.

On the *ECG* dataset the goal is to find patterns that predict the occurrence of a heartbeat. Our dataset is based on the first record (id 300.1) of the MIT-BIH ST Change Database.[3] We subsampled the record, replacing each 5 subsequent values with their average, transformed the result into a relative sequence by replacing each value with the difference to the previous value. Finally, using SAX [21] we discretize the sequence to 3 symbols. The heartbeats are annotated in the data. We shift the annotation slightly forward such that they are strictly before the heartbeat.

[3]https://physionet.org/content/stdb/1.0.0/

When we run it on this data, SQS discovers 12 patterns out of which only one is predictive: it corresponds to the previous heartbeat. SCIS requires a window length, which we set to the approximate length of one cardiac cycle, excluding the heartbeat ($w = 40$), for which it then returns 41 318 patterns. OMEN needs 388 seconds to discover two predictive patterns that together compress Y to only 70% of the number of bits needed by the null model. The first pattern corresponds to the previous heartbeat. The second, more subtle, pattern corresponds to the p-wave that occurs before a heartbeat. We visualized this pattern in Figure 6 and show the reported delay distribution. This experiment shows that OMEN is able to find non trivial patterns and delay distributions in real data.

Next we consider *Lifelog*, which is based on the life of *Sacha Chua* who logs and publishes all her daily activities.[4] We considered the data over 2017, removing any activities with have the same start and stop timestamp. As this dataset provides many events that are potentially interesting, we consider every $e \in \Omega$ as target, and have 40 target sequences with $Y[i] = 1$ iff $X[i] = e$. In addition, we consider a Y where we marked all business related activities as interesting.

Over all these datasets, SCIS discovers on average 695 patterns, many of which are redundant and not all make intuitive sense. While SQS only discovers 3 predictive patterns, these do make sense: *Cook, Dinner→Clean the Kitchen* and *Subway, Social→Subway*. OMEN takes between 6.1 and 37 seconds per dataset, and overall discovers 24 patterns. Many of these, such as *Sleep→Childcare*, *Cook→Dinner*, *Dinner→Clean the Kitchen*, predict the next action, i.e. a time delay distribution with a peak at 1. A more interesting pattern is *Subway→Subway* which has its peak at $\delta = 2$, and for which a natural interpretation is that *Sacha* takes the subway, logs on average one activity, and then takes the subway back.

Finally, we consider the *Saar* dataset, where the goal is to use daily precipitation records[5] to explain the rise (*Saar-Rise*) or fall (*Saar-Fall*) of the Saar river[6] by 10cm or more over one day. We considered the timespan from 2007 to 2018. We discretize the values to 17 symbols using $\lceil \log_{1.25} 1+x \rceil$ where we accumulate all values ≥ 15 into one symbol.

For SCIS with $w = 10$, we discover more than 400 patterns for either dataset. Many of these patterns are non-intuitive, such as that two successive days without rain predict a *rise* in water level. OMEN terminated for both datasets in under 4 seconds. SQS does not discover any descriptive nor predictive patterns. OMEN only reports singletons. Unsurprisingly, from the reported time delay distributions, we find that the more it rains the more likely it is for the Saar to have risen by 10cm or more, by either the next day or even two days later. For *Saar-Fall* we find an interesting pattern that expresses that approximately three days after heavy rain the water levels quickly drop—which indeed is likely as the water levels first rose due to rain.

[4]http://quantifiedawesome.com/records

[5]https://www.dwd.de/DE/leistungen/klimadatendeutschland/klarchivtagmonat.html (Ensheim weather station)

[6]Measured at Sankt Arnual by the Federal Institute of Hydrology (BfG)

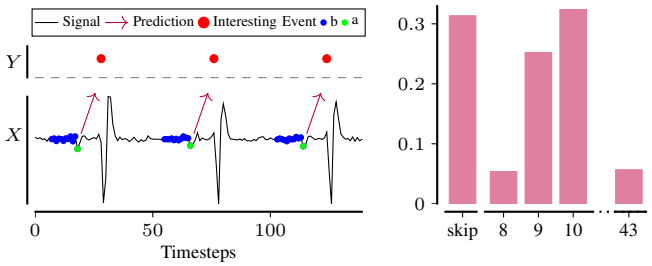

Figure 6: Window of ecg record with pattern *bbbbbbbbbbba* overlayed (left) and the reported time delay distribution (right).

VII. DISCUSSION

The experiments showed that OMEN works well in practice. We saw that its score is very good at telling predictive from spurious patterns, and that the mining algorithm is able to reconstructs the ground truth without picking up spurious or redundant patterns. In experiments on synthetic data we observed that it's highly robust against noise, and vastly outperforms the state of the art. On real world data we showed that it discovers easily interpretable patterns that reliably explain our target events. The results of the *ECG* experiment demonstrate that OMEN can find subtle patterns (p-wave pattern) even in the presents of clearer patterns (previous heartbeat). In summary, on all considered settings it discovers small, easily interpretable and non-redundant sets of reliable patterns that together predict the interesting events well.

We consider mining predictive rather than causal patterns. We do note, however, the close kinship between the two, and the fact that we share at least one common assumption: a cause needs to precede the effect in time [13], [23]. It will be interesting to explore under which additional conditions (assumptions) our patterns are indeed causal. By encoding Y using delay distributions that are independent of the actual pattern occurrences in X, our framework naturally fits the algorithmic model of causality [17], which could explain why our score performs so well in comparison to CUTE [5] and TENT [28]. By explicitly maximizing this independence we could possibly adapt OMEN to discover sequential patterns that are causal under the additive noise model [29], [16], which is an interesting direction for future work.

Although OMEN is effective at discovering meaningful patterns, it currently considers a relatively simple pattern language. At the expense of additional computation, it is straightforward to extend both the OMEN score and search to sequences with gaps, serial episodes [18] or complex multivariate patterns [3].

Even though OMEN works well in practice, its alignment algorithm allows for further improvement. While it currently fails for adversarial input, at the expense of computation we could compute optimal alignments via dynamic programming [30]. A more interesting extension would be to include prior knowledge on the time-delay distributions, such as a shape (Gaussian, Poisson) or expected delay.

				SCIS	SQS +OMEN	OMEN		
Dataset	$\|X\|$	$\|\Omega\|$	$\|\|Y\|\|_1$	Found	$\|M\|$	$\|M\|$	$\%L$	Runtime in sec
ECG	107395	3	2558	41318	1	2	70	388
Saar-Rise	4018	17	278	419	0	7	78	4
Saar-Fall	4018	17	278	442	0	2	98.5	4
Lifelog	5970	40	153	695	0.07	0.6	95	6

Table I: Results on real datasets. We give data sequence length, alphabet size and number of interesting events in Y and the number of reported patterns for SQS + OMEN and SCIS. For OMEN we additional report compression rate relative to the null model in percent, $\%L$ and runtime. For *Lifelog* we report the average over 41 independent runs, with different target events.

VIII. CONCLUSION

We considered the problem of discovering small sets of sequential patterns that not only predict that something interesting will happen, but for which it is additionally easy to tell how long it will be until the predicted event. We formulated the problem in information-theoretic terms using the Minimum Description Length principle. As the resulting problem does not lend itself of efficient exact optimization, we propose the OMEN algorithm to heuristically discover good alignments and pattern sets. Extensive evaluation on synthetic and real world data showed that OMEN compares favorably to the state of the art. In particular, the OMEN score performs very well in telling predictive from associative patterns, even under large quantities of noise. The OMEN pattern miner efficiently discovers high quality sets of predictive patterns give clear insight into the data generating process. In future work we will focus on generalizing the pattern language, as well as investigating under which conditions we can discover sequential patterns that are causal.

REFERENCES

[1] C. C. Aggarwal and J. Han, Eds., *Frequent Pattern Mining*. Springer, 2004.

[2] I. Batal, G. F. Cooper, D. Fradkin, J. Harrison, F. Moerchen, and M. Hauskrecht, "An efficient pattern mining approach for event detection in multivariate temporal data," *Knowl. Inf. Sys.*, vol. 46, no. 1, pp. 115–150, 2016.

[3] R. Bertens, J. Vreeken, and A. Siebes, "Keeping it short and simple: Summarising complex event sequences with multivariate patterns," in *KDD*, 2016, pp. 735–744.

[4] A. Bhattacharyya and J. Vreeken, "Efficiently summarising event sequences with rich interleaving patterns," in *SDM*. SIAM, 2017, pp. 795–803.

[5] K. Budhathoki and J. Vreeken, "Causal inference on event sequences," in *SDM*. SIAM, 2018, pp. 55–63.

[6] Y. Chen, G. Rangarajan, J. Feng, and M. Ding, "Analyzing multiple nonlinear time series with extended granger causality," *Phys. Lett. A*, vol. 324, no. 1, pp. 26–35, 2004.

[7] C. Corbière, N. Thome, A. Bar-Hen, M. Cord, and P. Pérez, "Addressing failure prediction by learning model confidence," in *NIPS*, 2019, pp. 2898–2909.

[8] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley-Interscience New York, 2006.

[9] E. Egho, D. Gay, M. Boullé, N. Voisine, and F. Clérot, "A user parameter-free approach for mining robust sequential classification rules," *Knowl. Inf. Sys.*, vol. 52, no. 1, pp. 53–81, 2017.

[10] J. Fowkes and C. Sutton, "A subsequence interleaving model for sequential pattern mining," in *KDD*, 2016, pp. 835–844.

[11] E. Galbrun, P. Cellier, N. Tatti, A. Termier, and B. Crémilleux, "Mining periodic patterns with a mdl criterion," in *ECML PKDD*. Springer, 2018, pp. 535–551.

[12] M. S. Gerber, "Predicting crime using twitter and kernel density estimation," *Decision Support Systems*, vol. 61, pp. 115–125, 2014.

[13] C. W. Granger, "Investigating causal relations by econometric models and cross-spectral methods," *Econometrica*, pp. 424–438, 1969.

[14] P. D. Grünwald and A. Grunwald, *The minimum description length principle*. MIT press, 2007.

[15] B. Hooi and C. Faloutsos, "Branch and border: Partition-based change detection in multivariate time series," in *SDM*. SIAM, 2019, pp. 504–512.

[16] P. Hoyer, D. Janzing, J. Mooij, J. Peters, and B. Schölkopf, "Nonlinear causal discovery with additive noise models," 2009, pp. 689–696.

[17] D. Janzing and B. Scholkopf, "Causal inference using the algorithmic markov condition," *IEEE TIT*, vol. 56, no. 10, pp. 5168–5194, 2010.

[18] S. Laxman, P. S. Sastry, and K. P. Unnikrishnan, "A fast algorithm for finding frequent episodes in event streams," in *KDD*. ACM, 2007, pp. 410–419.

[19] S. Laxman, V. Tankasali, and R. W. White, "Stream prediction using a generative model based on frequent episodes in event sequences," in *KDD*, 2008, pp. 453–461.

[20] M. Li and P. Vitányi, *An Introduction to Kolmogorov Complexity and its Applications*. Springer, 1993.

[21] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing sax: a novel symbolic representation of time series," *Data Min. Knowl. Disc.*, vol. 15, no. 2, pp. 107–144, 2007.

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

[23] J. Pearl, *Causality: Models, Reasoning and Inference*, 2nd ed. Cambridge University Press, 2009.

[24] K. Radinsky and E. Horvitz, "Mining the web to predict future events," in *WSDM*, 2013, pp. 255–264.

[25] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.

[26] ——, "A universal prior for integers and estimation by minimum description length," *Annals Stat.*, vol. 11, no. 2, pp. 416–431, 1983.

[27] E. Scharwächter and E. Müller, "Two-sample testing for event impacts in time series," in *SDM*. SIAM, 2020, pp. 10–18.

[28] T. Schreiber, "Measuring information transfer," *Phys. Rev. Lett.*, vol. 85, no. 2, p. 461, 2000.

[29] S. Shimizu, P. O. Hoyer, A. Hyvärinen, and A. Kerminen, "A linear non-gaussian acyclic model for causal discovery," *JMLR*, vol. 7, pp. 2003–2030, 2006.

[30] N. Tatti and J. Vreeken, "The long and the short of it: summarising event sequences with serial episodes," in *KDD*, 2012, pp. 462–470.

[31] J. Wang, J. Han, and C. Li, "Frequent closed sequence mining without candidate maintenance," *IEEE TKDE*, vol. 19, no. 8, pp. 1042–1056, 2007.

[32] G. M. Weiss and H. Hirsh, "Learning to predict rare events in event sequences." in *KDD*, vol. 98, 1998, pp. 359–363.

[33] Q. Wu, Y. Gao, X. Gao, P. Weng, and G. Chen, "Dual sequential prediction models linking sequential recommendation and information dissemination," in *KDD*, 2019, pp. 447–457.

[34] L. Zhao, J. Ye, F. Chen, C.-T. Lu, and N. Ramakrishnan, "Hierarchical incomplete multi-source feature learning for spatiotemporal event forecasting," in *KDD*, 2016, pp. 2085–2094.

[35] C. Zhou, B. Cule, and B. Goethals, "Pattern based sequence classification," *IEEE TKDE*, vol. 28, no. 5, pp. 1285–1298, 2016.

[36] C. Zhou, B. Cule, and B. Goethals, "A pattern based predictor for event streams," *Expert Syst. Appl.*, vol. 42, no. 23, pp. 9294–9306, 2015.

APPENDIX

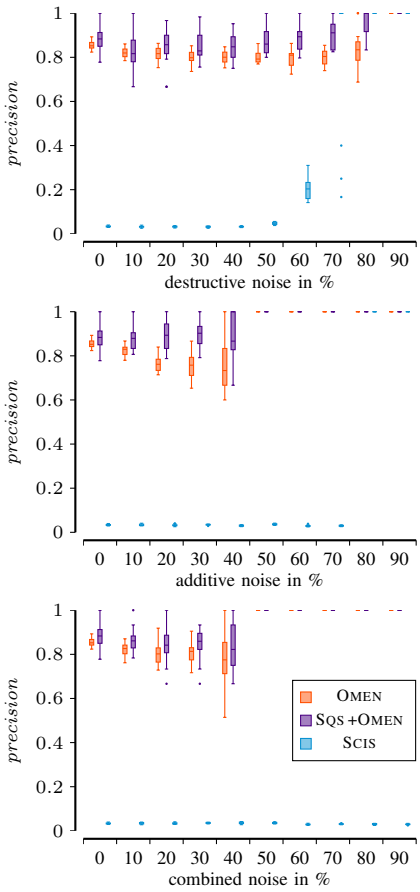

Figure 7: [Higher is better] Precision results on synthetic data for destructive (left), additive (middle) and combined (right) noise.

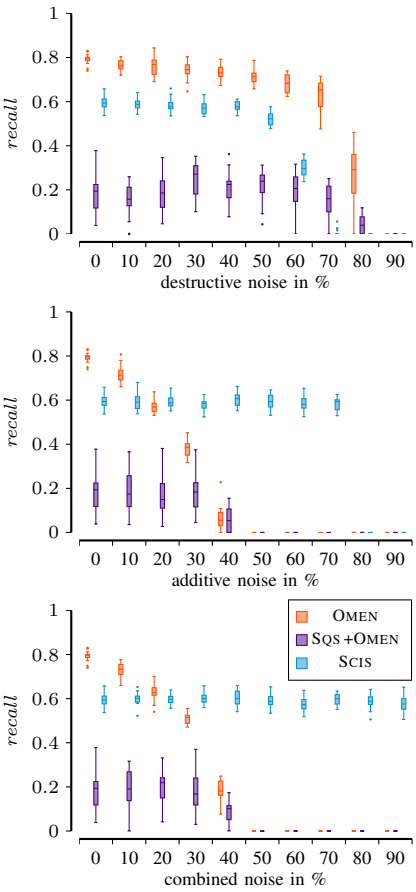

Figure 8: [Higher is better] Recall results on synthetic data for destructive (left), additive (middle) and combined (right) noise.