# Predicting Target Events in Industrial Domains

Julio Borges[1(✉)], Martin A. Neumann[1], Christian Bauer[2], Yong Ding[1], Till Riedel[1], and Michael Beigl[1]

[1] TECO, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
{julio.borges,martin.neumann,yong.ding,till.riedel,michael.beigl}@kit.edu
[2] TRUMPF Werkzeugmaschinen GmbH + Co. KG, Ditzingen, Germany
Christian.Bauer@de.TRUMPF.com

**Abstract.** In industrial environments, machine faults have a high impact on productivity due to the high costs it can cause. Machine generated event logs are a abundant source of information for understanding the causes and events that led to a critical event in the machine. In this work, we present a Sequence-Mining based technique to automatically extract sequential patterns of events from machine log data for understanding and predicting machine critical events. By experiments using real data with millions of log entries from over 150 industrial computer numerical control (CNC) cutting machines, we show how our technique can be successfully used for understanding the root causes of certain critical events, as well as for building monitors for predicting them long before they happen, outperforming existing techniques.

## 1 Introduction

In today's industry, the success of manufacturing companies highly depends on reliability and quality of their machines and products for their production process. Unexpected machine failures in production processes are bounded to high repair costs and production delays [1]. Therefore, understanding and predicting critical situations before they occur can be a valuable source for avoiding unexpected breakdowns and saving costs associated to the failure.

Log-files keep a record on the flow of states and activities performed by the machine presenting therefore an important source of information on how a machine "behave" prior to a critical situation [2]. Understanding what are the causes that lead to specific critical situations in a machine can help engineering and maintenance teams to build a problem diagnosis and repair the machine. In many industries, this task is still bounded to high manual efforts, as the maintenance staff must go through the data in order to evaluate what are the possible causes for critical events, demanding expert domain knowledge of the system. Automatically diagnosing and even predicting when critical events are about to occur can thus represent a significant advantage in the industry for both **critical event diagnosis**, i.e., understanding what caused a critical event, to **critical event prediction**, monitoring and alarming when a critical event is about to occur even before it happens, enabling pre-intrusion to avoid system problems.

P. Perner (Ed.): MLDM 2017, LNAI 10358, pp. 17–31, 2017.
DOI: 10.1007/978-3-319-62416-7_2

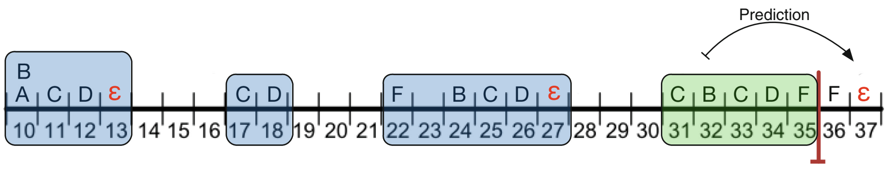

**Fig. 1.** Event-Based Prediction based on Sequence Mining. The sequential rule "$\langle B, C, D\rangle \Rightarrow \varepsilon$" describes the sequence of 3 events that always appear in sequence before the target (critical) event ($\varepsilon$). This information can be used for monitoring and predicting interesting target events in advance.

This paper focuses on the latter. This requires some short-term anticipation of upcoming critical event based on evaluation of the running state of the system.

The critical event prediction approach to be introduced by this paper is based on the batch offline analysis of logged error events. Frequent patterns of sequences of events that lead to errors are mined from the logs used to construct a monitor which can posteriorly observe events during runtime. Upcoming target critical events are then predicted by monitoring the events that have occurred recently before present time (cf. Fig. 1). Experiments on industrial data of industrial computer numerical control (CNC) cutting machines have shown superior prediction performance in comparison with the most well-known event-based prediction algorithms in that area.

The rest of this paper is organized as follows. Section 2 presents related work in the area of event based predictive maintenance. Section 3 describes the underlying collected dataset used for our experiments in this work. Section 4 presents the necessary data preparation steps for our algorithm. Section 5 presents our proposed algorithm which is evaluated and compared against the state of the art in Sect. 6. Section 7 then concludes the paper.

## 2 Related Work

In light of *Industry 4.0*, intelligent analysis of machine generated data specially for the task of predictive maintenance has gained increased attention recently.

`Salfner et al.` published a broad survey on the application of log analysis for what they call short-term **online failure prediction**. The goal is to predict the occurrence of failures during runtime based on the current system state [3]. They proposed in [2] an online failure prediction technique based on log files by using hidden semi-Markov predictor (HSMM) and appropriate pre-processing of the data. Two HSMMs are trained from previously recorded log data: one for failure and one for non-target sequences. Online failure prediction is then accomplished by computing likelihood of the observed error sequence for both models and by applying Bayes decision theory to classify the sequence (and hence the current system status) as failure-prone or not. They evaluated their approach on commercial telecommunication system data reporting failure prediction F-Measure up to 0.66.

`Vilalta et al.` proposed a prediction technique by applying itemset mining on a set of events preceding a target event on computer network data [4]. The dataset is partitioned in failure and non-failure event sets w.r.t. a target event. Events preceding the target event in a time window constitute the failure event sets. `Vilalta et al.` then applies association rule mining for extracting frequent and valid association rules on the partitioned dataset. In this process, the failure event sets are checked against frequency (support test) and validity (confidence). In contrast to sequence mining techniques (as this work is based upon), the event sets are unordered, so the chronological sequence of events plays no role in the training process. Consequently, our work considers the chronological order of occurring events. `Vilalta et al.` reported that depending on the target event, the prediction accuracy of their approach can significantly vary. Meanwhile they observed that for an event they had a false negative error rate of only around 4.5%, for another target event the error rate was so high as 83% with the same algorithm parametrization. We have made similar observations which will be discussed in Sect. 6.2.

## 3 Data Description

The log datasets used in this study derives from a real production scenario of a industrial computer numerical control (CNC) cutting machine system from the company TRUMPF GmbH + Co. KG, one of the world's biggest providers of machine tools. Overall, we collected big log datasets from 154 unique machines. Altogether, there are more than 4 million logged events in the datasets.

A **log dataset** is a collection of **events** recorded by various applications/-components running on the machine (see Table 1). An event in this setup consists mainly of an unique and representative event code indicating an entering state or action being performed by the machine (ex.: Machine door opened), a timestamp indicating when the event occurred and an event severity class (Info, Warning, Error, etc.). They can have a systematic nature or be caused by an human intervention on the machine. Events reflect the developers' original idea about what are the valuable states to report.

**Table 1.** A piece of a real log file (with renamed event codes and messages for Blind Review) from a computer numerical control (CNC) cutting machine

| Timestamp | Eventcode | Severity | Message |
|---|---|---|---|
| 12/22/2014 07:24:23 | 300 | Info | Data transfer 12345 |
| 12/22/2014 07:57:49 | 600 | Warning | Check *XX* level |
| 12/22/2014 08:01:28 | 900 | Error | Error in *YY* system |

In our application, we have altogether more than 1400 unique event codes from different categories. Although our approach can be used to predict any arbitrary target event, we focus particurlaly on predicting events which signalize critical machine events and situations.

## 4 Data Modeling

The first challenge that arises when applying Sequence Mining is how to proper model the raw machine log data into a sequence representation for the sequence miner to find patterns on. This section describes the necessary steps to transform the raw event logs into proper temporal event sequences used as input for the model to be proposed by this paper - while comparing to other modeling methods presented in the literature.

### 4.1 Static Time Window

Time window based approaches are techniques that group all events logs preceding a target event within a given time interval (the window size, cf. [4]). All grouped events within the time window are called a **target sequence**. i.e., a target sequence is a positive case containing the target critical event. **Non-Target sequences** denote sequences that have occurred between target sequences: starting at the beginning of the sequence, non-overlapping time windows of size $W$ that do not intersect the set of time windows preceding target events (target sequences) are considered negative cases.

A big drawback of such strategies for data modeling is caused by the use of grids (partitioning the data in equidistant time intervals). In general, grid-based approaches heavily depend on the positioning of the grids: sequential patterns may be missed if they are inadequately oriented or shaped. Therefore, alternative data modeling strategies are desired.

### 4.2 Dynamic Time-Window: The Clustering Strategy

The clustering/tupling strategy for grouping events in a log file has been first presented by `Tsao et al.` in [5] and further discussed and analyzed by `Hansen et al.` in [6]. Tupling basically refers to grouping (*clustering*) events that occur within a pre-defined time interval in the log. The intuition underlying this approach is that two entries in the log, if related, are likely to occur near in time [6]. Consequently, if their inter-arrival time distance is below a predetermined threshold $\Delta t$ (called the *coalescence window*), they are placed in the same group (called *tuple*). `Lal and Choi` have shown that the tupling method can also be used for clustering events that are related to the same target category in time [7]. They have reported that the clusters can successfully gather bursts of events in the machine logs. This is formalized in Definition 1.

**Definition 1.** ***Cluster Based Sequence:*** *given are $n$ timestamped, chronologically ordered log entries $e_1, ..., e_n$. Let $t_i$ stands for the time of occurrence of event $e_i$ and $\Delta t$ be a user specified threshold parameter. Then a sequence $S$ in a sequence database SDB is represented by $S = \{\langle e_i, ..., e_k\rangle | i, k \in [1, n], i < k : t_{j+1} - t_j < \Delta t, j \in [i, k-1]\}$.*

After this step (which is independent of the target critical event to be predicted), a label definition of a **target** and **non-target** sequence (cf. Sect. 4.2) must be provided in order to specify which events in the sequences of the database will serve as possible patterns for pattern recognition techniques to act upon. Given an target critical event $\varepsilon$, the set of target sequences (positive cases) is a subset of the sequence database SDB which contains the target critical event $\varepsilon$. i.e., the set of target sequences for a target critical event $\varepsilon = \{S \in SDB | \varepsilon \in S\}$. Only events which appears prior to the appearance of the target event in the target sequence are relevant to be analyzed as the cause of possible critical situations This is defined in Definition 2 and shown in Fig. 2.

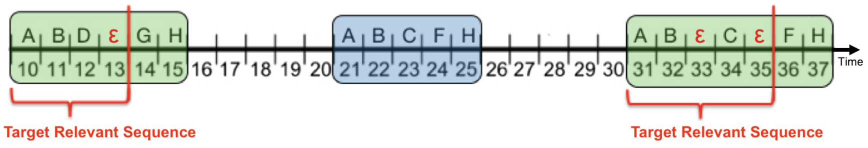

**Fig. 2.** For a given target critical target event $\varepsilon$, target and non-target clusters are depicted, highlighting target relevant sequences.

**Definition 2.** ***Target Relevant Sequences***[1]*: A sequence set $SDB_\varepsilon$ of Target Relevant Sequences for a target critical event $\varepsilon$ is a subset of subsequences from a sequence database SDB defined as:*

$$SDB_\varepsilon = \{S' | S' \sqsubseteq S, S \in SDB\}, S' = \langle e_i, .., e_j \rangle, e_j = \varepsilon, \nexists k > j : e_k = \varepsilon \wedge e_k \in S.$$

Concluding this section, static time windows are sensitive to window resolution and position. Clustering based-approaches on the other side are position-independent and only depend on the resolution (coalescence window). Since critical events are rather rare, our approach just searches for patterns within target relevant sequences, dramatically reducing the search space.

## 5 Approach

Sequence Mining is a data mining technique that focuses on finding statistically relevant patterns in a sequence form for a given data set. We propose an approach to extract sequential patterns from temporal data to predict the occurrence of target events, such as critical situations logged by a machine. The first step in order to apply sequence mining to log data is transformation of the log into a proper sequence representation for the sequence miner to find patterns on, as discussed in the last section. In this section, we propose a prediction technique based on transforming the event prediction problem into a search for all frequent and valid sequences preceding a target event. Patterns are then integrated into a rule-based model for prediction.

[1] We use the notation '$\sqsubset$' to denote subsequence and '$\subset$' to denote subset relations.

### 5.1 Mining Sequential Patterns

A sequence in our context denotes a chronological ordered list of itemsets (each item is a machine event). The support (frequency) of a sequence $S_j$ in a sequence database SDB is defined as the portion of sequences $S \in SDB$, such that $S_j \sqsubseteq S$. Let $min_{sup}$ be a threshold on the support value: a sequence $S_j$ is deemed frequent iff $support(S_j) > min_{sup}$. The problem of mining sequential patterns is to discover all frequent sequences. For our implementation in this work, we utilize the algorithm SPADE [8]. SPADE is an efficient algorithm that decomposes the original problem of finding frequent patterns into smaller subproblems which can be solved independently using efficient lattice search techniques and join operations. Nonetheless, several algorithms featuring different properties have been proposed in the literature for this task [9].

**Mining Sequential Rules.** Once the frequent sequences are known, they can be used to obtain rules that describe the relationship between different sequence events. We are particularly interested if it is possible to infer the probability of appearance of a target critical event ($\varepsilon$) given an observed previous sequence of events. To solve this kind of problem, using sequence mining by mining frequent sequences alone is very limited. For example, consider the pattern $S = \langle X, \varepsilon \rangle$, which means that it is possible that $\varepsilon$ appears frequently after an observed sequence $X$. In this case, we talk about a *sequential rule*, denoted by $X \Rightarrow \varepsilon$, derived from $\langle X, \varepsilon \rangle$.

**Definition 3.** *$\varepsilon$-**Rule:** a $\varepsilon$-sequential-rule, for a target event $\varepsilon$, is a sequence in the form $R = \langle X, \varepsilon \rangle$, whereby $X$ is a sequence in $R$ preceding the target event $\varepsilon$. In this case we write: $R = \langle X, \varepsilon \rangle = (\langle X \rangle \Rightarrow \varepsilon) = (X \Rightarrow \varepsilon)$.*

The challenge is: there may be also many cases where $X$ is not followed by $\varepsilon$ in SDB. For prediction, we need a measurement of the conditional probability, that if $X$ occurs, $\varepsilon$ will also occur afterwards. The measure is the **confidence** defined as:

$$confidence_{SDB}(X \Rightarrow \varepsilon) = P(\varepsilon \mid X) = \frac{support_{SDB}(\langle X, \varepsilon \rangle)}{support_{SDB}(\langle X \rangle)} \tag{1}$$

The higher the confidence, the higher the probability, that $\varepsilon$ occurs **after** $X$. Given a user-specified minimum confidence ($min_{conf}$), sequential mining algorithms extract all rules which fulfills the condition, i.e., those that comply to the minimum desired confidence. Note that due to the apriori property, for a frequent sequence $S = \langle X, \varepsilon \rangle$, all the subsequences of $X$ are also frequent, also those which do not contain $\varepsilon$. For this reason, they are filtered out of the output from the sequence miner. We call this result set the $\varepsilon$-Rules Data Base $RDB_\varepsilon$ (cf. Algorithm 1). Sequence mining can thus be leveraged for creating sequential rules delivering the probability of occurrence of an critical situation in the machine given a previously observed sequence.

### 5.2 Filtering the Noise

Machinery logs often contain events that are unrelated to a specific critical target event, but due to different processes logging in the machine simultaneously, the target event may be interlaced with unrelated events. Applying Sequence Mining techniques for the task of predicting future critical situations can deliver many sequential patterns from which many of them may be redundant and have no stronger predictive power. The apriori property states: if a sequence $S = \langle X, \varepsilon \rangle$ is frequent w.r.t. $min_{sup}$, so is every sequence $S'$ with $S' \sqsubseteq S$. The Sequence Mining algorithm will thus output all subsequences of $X$ for every closed sequence S, as a possible explanation to $\varepsilon$ if they pass the confidence test.

Consequently, some post-processing noise remotion techniques become necessary in order to remove noisy, unpredictive and redundant patterns which reduce the size of the returned rule set. In the following, some definitions will be presented which will be necessary for presenting the post-processing noise remotion strategy.

**Definition 4.** ***Confidence Gain:*** *given two sequential rules $R_i$, $R_j \in RDB_\varepsilon$ with $R_i \sqsubset R_j$. The confidence gain of $R_j$ w.r.t. $R_i$ is defined as:*

$$confidence_{gain}(R_j, R_i) = confidence_{SDB}(R_j) - confidence_{SDB}(R_i) \qquad (2)$$

With the confidence gain, we measure the increase in predictive information gained by augmenting the left-hand of a rule $R_i$ to an augmented rule $R_j$. We identify a sequential rule as redundant in $RDB_\varepsilon$ if it delivers no gain in confidence. Formally:

**Definition 5.** ***Redundancy-Free Rule:*** *given a sequential rule $R_j \in RDB_\varepsilon$. The rule $R_j$ is deemed redundancy-free iff:*

$$\forall R_i \sqsubset R_j, R_i \in RDB_\varepsilon : confidence_{gain}(R_j, R_i) > 0 \qquad (3)$$

*Otherwise it is deemed* ***redundant****.*

If a rule is not redundancy-free, it is deemed redundant and it is filtered out as noise in our approach. Consequently, a rule $R_j$ is deemed redundant, if there exists another rule $R_i$ with: $R_i \sqsubset R_j$ and $confidence(R_i) = confidence(R_j)$. Due to the anti-monotonicity property of the support constraint, $R_i$ is at least as frequent as $R_j$ [8]. Consequently, we eliminate all redundant sequences that are **not** more predictive than any of their proper subsequences.

The rationale is that, sequential patterns can be augmented with noisy events which do not contribute to a increase in confidence gain at predicting the target event $\varepsilon$. Take for example $R_1 := \langle e_1, e_2, e_3, n \rangle \Rightarrow \varepsilon$ as a rule with high confidence and with a noise event $n$. In this case, the event $n$ does not contribute to a better predicability when compared to its subsequence rule $R_2 := \langle e_1, e_2, e_3 \rangle \Rightarrow \varepsilon$. And in this case the prediction lead time of $R_2$ is per definition better that the prediction lead time of $R_1$, as $n$ comes after $e_3$ as last element in the rule. $R_1$ is thus a redundant rule and can be removed from the result set.

In case $R_j$ is a redundancy-free rule, we keep $R_j$ and eliminate all those $R_i$, with $R_i \sqsubset R_j$. I.e., we eliminate all sequences, which are less predictive than any of its proper super-sequences. This leads to the concept of a *redundancy-free sequence set*:

**Definition 6.** ***Redundancy-Free Sequence Set:*** *given a sequence set $\mathcal{KB} \subseteq RDB_\varepsilon$, $\mathcal{KB}$ is a Redundancy-Free Sequence Set if:*

$$\forall R \in \mathcal{KB} : R \; \texttt{is redundancy-free} \tag{4}$$

We still must guarantee, that we output all possible redundancy-free sequential rules from $RDB_\varepsilon$. For this, we introduce the concept of *concept-covering sequence set*. Intuitively, a set of rules $\mathcal{KB}$ is concept-covering if adding any new rule $\in RDB_\varepsilon$ into $\mathcal{KB}$ always results in redundancy.

**Definition 7.** ***Concept-Covering Sequence Set:*** *given a sequence set $\mathcal{KB} \subseteq RDB_\varepsilon$, $\mathcal{KB}$ is concept-covering if:*

$$\forall \overline{R} \in RDB_\varepsilon \backslash \mathcal{KB} : \overline{R} \; \texttt{is not redundancy-free} \tag{5}$$

In our definition $\mathcal{KB}$ must be redundancy-free and concept covering. We output $\mathcal{KB}$ as *knowledge base* instead of $RDB_\varepsilon$, which contains just a small set of high confident and non-redundant rules. The knowledge base $\mathcal{KB}$ is guaranteed to contain only rules which do not contain noisy events w.r.t. Definition 4.

### 5.3 Building a Sequence-Based Predictive Model

The resulting redundancy-free and concept-covering set of rules $\mathcal{KB}$ (Definition 7) can be leveraged to build the so called predictive ***monitor***.

**Definition 8.** ***Monitor:*** *a function that takes as input a sequence $S$ and outputs a predicted event if any of the rules in a knowledge base $\mathcal{KB}$ matches the input sequence:*

$$Monitor(S) := \{\varepsilon \mid R \in \mathcal{KB}, R = (X \Rightarrow \varepsilon) : X \sqsubseteq S\} \tag{6}$$

Please note that due to the filtering of redundant rules (Definition 5) the monitor guarantees that no sequences which may be subsequences from each other can match an input sequence, avoiding redundant matches.

Let's take as example a monitor $M_R$ with a single rule $R := \langle A, B\rangle \Rightarrow E$. The monitor must make the decision (prediction) to fire or not an alarm while observing the sequence of logged events of the machine for the target critical event. This is a typical binary decision case where each monitor's decision can fall into the following 4 categories:

- **TP - True-Positive (Hit):** an alert/alarm occurs and the target critical event is observed in the given sequence after the alert. The sequences $\langle A, B, C, E\rangle$ and $\langle E, A, B, C, E\rangle$ for example are counted as hit for $M_R$.

**Algorithm 1.** Sequence Based Failure Predictor (SBFP)

```
1: procedure SBFP(Target Event $\varepsilon$, Input Data $Logs$, $min_{sup}$, $min_{conf}$, $\Delta t$)
2:     SDB ← TemporalCluster($Logs$, $\Delta t$)                      ▷ Cluster log files w.r.t. Definition 1
3:     $SDB_\varepsilon$ ← FRS($\varepsilon$, SDB)          ▷ Extract Failure Relevant Sequences w.r.t. Definition 2
4:     $\mathcal{FS} \leftarrow SequenceMiner(SDB_\varepsilon, min_{sup})$                 ▷ Sequence Miner, e.g. SPADE
5:     $RDB_\varepsilon \leftarrow SET()$
6:     for all $S \in \mathcal{FS}$ do                              ▷ Filter valid $\varepsilon$-Rules w.r.t. Definition 3
7:         if $S$ is $\varepsilon$-Rule AND $confidence_{SDB}(S) \geq min_{conf}$ then
8:             $RDB_\varepsilon.add(S)$
9:         end if
10:    end for
11:    $\mathcal{KB}$ ← redundancy-free and concept covering subset from $RDB_\varepsilon$ w.r.t. Definitions 6 and 7
12:    return Monitor($\mathcal{KB}$)          ▷ Build Monitor with Knowledge Base $\mathcal{KB}$ w.r.t. Definition 8
13: end procedure
```

- **FP - False-Positive:** false alarms. E.g. $\langle A, B, C\rangle$ and $\langle E, A, B, C\rangle$ for $M_R$.
- **TN - True-Negative:** no alarm is raised and the observed log sequence does not contain the target event. E.g. $\langle B, A, C\rangle$ for $M_R$.
- **FN - False-Negative (Miss):** no alarm has been raised but the target event was observed in the sequence. E.g. $\langle A, C, E\rangle$ for $M_R$.

Based on these premises, an intensive evaluation of the presented rule-based predictive model will be performed and discussed in the next section. The technique is summarized in Algorithm 1.

**Computational Efficiency.** Looking for all frequent eventsets is in the worst case exponential in the number of single event types. Note that the Algorithm 1 (line 4) only scans $SDB_\varepsilon$ and not the sequence database $SDB$ itself in search for sequential patterns. Since the number of sequences containing target events is much lower than the total number of sequences, we scan a much smaller projection of SDB in initial steps guaranteeing inexpensive computational efficiency in both memory and time in first steps. However, the complexity of the approach is highly dependent on the complexity of the sequence mining algorithm chosen in this step. In our case, we deploy the SPADE algorithm, which features a linear scalability w.r.t. the number of sequences [10]. Finally, the line 7 of the algorithm filter out all frequent sequences which do not contain the target event $\varepsilon$ or are invalid w.r.t. minimum confidence $min_{conf}$.

## 6 Evaluation

We carry out extensive experiments with the proposed predictive model, leveraging real world log datasets from industrial computer numerical control (CNC) cutting machine systems (cf. Sect. 3). The performance of the proposed approach is then compared to the state-of-the-art method in event prediction proposed by `Vilalta et al.` [4]. As a reminder, `Vilalta et al.` proposed a method based on time window modeling of the log data (cf. Sect. 4.1), where they apply temporal itemset mining to extract association rules for predicting target critical events. Please note that the underlying assumption of the method proposed by `Vilalta et al.` is that the chronological sequence in which the events appear prior to a critical event is not significant. Just the set of logged events, independent of the sequence in which they appear in the log, are leveraged by `Vilalta et al.` for their prediction model. We claim that the set of events alone is not as powerful as the actual chronological order of the events in the set (Sequence Mining) for our use-case.

We evaluate the prediction performance of a Monitor in a holdout setting using binary classification performance metrics such as the precision and the recall. The point of partition of the dataset is set to the 50%th-index of the target critical event in the log: log events to the left of this point are used for training and in the right for testing, assuring that both partitions contain the exact same occurence number of the target event in it and that the training data lies on the past of the test data. The precision of a monitor is the percentage of times the monitor signals a critical event, and the critical event actually occurs (i.e., the ratio of correct critical situation signals to the total number of critical situation signals). The recall of a monitor is the percentage of critical events signaled prior to their occurrence. We leverage the $F_\beta$-$Measure$ which balances the precision and the recall, as a target function to be optimized (maximized):

$$F_\beta\text{-}Measure = (1+\beta^2)\cdot\left(\frac{Precision \cdot Recall}{(\beta^2 \cdot Precision) + Recall}\right) \tag{7}$$

The $F_1$-Measure is the most widespread metric for comparing or evaluating classifiers. Two other commonly used F measures are the $F_2$-Measure, which weights recall higher than precision, and the $F_{0.5}$-Measure, which puts more emphasis on precision than recall. We argue, that for the use-case in question (predicting machine critical events), the $F_{0.5}$-Measure is more relevant, as the precision of the monitor/predictor is more important than the recall, i.e., misses are in our scenario not so bad as false alarms. This argument lies on the assumption that producing false alarms may generate *reaction costs*, so we prefer more conservative and confident models in opposite to more general ones. For this reason, we set the $F_{0.5}$-Measure as the main evaluation metric to be used in this work.

Given a target critical event, we are interested in answering following questions: How does the proposed approach perform at predicting the target critical event? Which parametrization leads to best prediction performance? Which prediction lead time can we expect for predicting the target event in our context?

These questions will be answered on top of experiments. Domain experts ranked the 13 most relevant target critical events to be analyzed in this work. We select the machine log dataset where the most important ranked target event most often occurs for a detailed evaluation in Sect. 6.1. The other remaining 12 target events are also be analyzed and evaluated across varying machine datasets in Sect. 6.2.

## 6.1 Parametrization and Prediction Performance

We optimize the 3 parameters of the proposed approach, namely the coalescence window ($\Delta t$), the $min_{sup}$ and $min_{conf}$ parameters through grid search [11] for the $F_{0.5}$-Measure as scorer function. In grid search the set of trials is formed by assembling every possible combination of values and the best parametrization w.r.t. scorer function is returned. The overall best parameter space is identified and used for training the end model. Our grid search yells as result the parameters $\Delta t = 450\,\text{s}$, $min_{sup} = 10\%$ and $min_{conf} = 70\%$ as the overall best parametrization of the model which optimizes the $F_{0.5}$-Measure.

Naturally, the minimum desired support and confidence parameters ($min_{sup}$ & $min_{conf}$) have a big impact on the prediction performance. The use of a low support can allow rare rules to pass the frequency test, but this comes at the cost of increasing the complexity of the search space generating excessively many rules. For our scenario, setting $min_{sup} = 0.1$ turned out to be optimal since mining more rules (by setting a lower $min_{sup}$) did not the enrich the knowledge base significantly (i.e., most new rules are redundant below this value w.r.t. confidence gain).

Figure 3 shows how the prediction performance behave in dependence of the confidence parameter. The minimum desired confidence ($min_{conf}$) is a trade-off between precision and recall. Rules with high confidence make predictions only with strong evidence so they lead to overall higher precision, but they often also lead to low recall rates.

Last but not least, the coalescence window also impact the prediction performance as well as the prediction lead time of the proposed algorithm. As can be seen on the Fig. 4 (left), all other parameters being equal, the prediction performance increases together with larger coalescence windows up to a certain point.

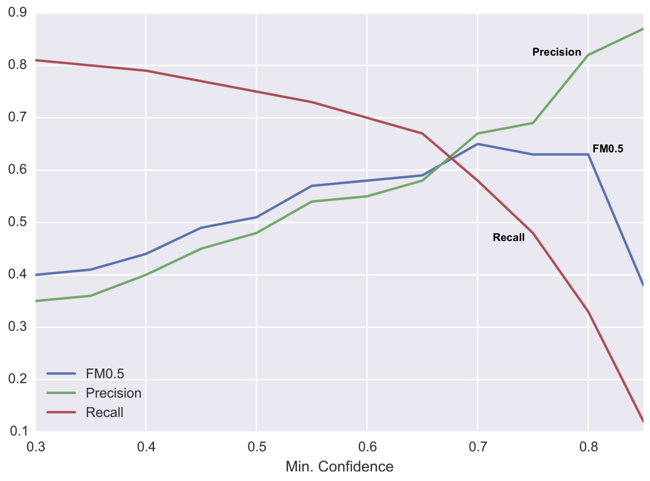

**Fig. 3.** Variation of prediction metrics depending on the minimum desired confidence parameter.

Larger coalescence windows enable us to capture more information preceding target events. But just until to the point where the number of collisions[2] causes several sequences to be merged, loosing contrast for separating the events which are related to the target critical event and those which are not.

Figure 4 (right) shows the distribution of the prediction lead time for the tested dataset. Prediction models which are based on modeling the log data with a static time window strategy are bounded to a maximum prediction lead time of the size of the window itself, which is not our case. Such a data modeling strategy enables a flexible short-term prediction lead time, which can be by orders of magnitude longer than the coalescence window parameter itself (up to 40 min), as shown in Fig. 4 (right).

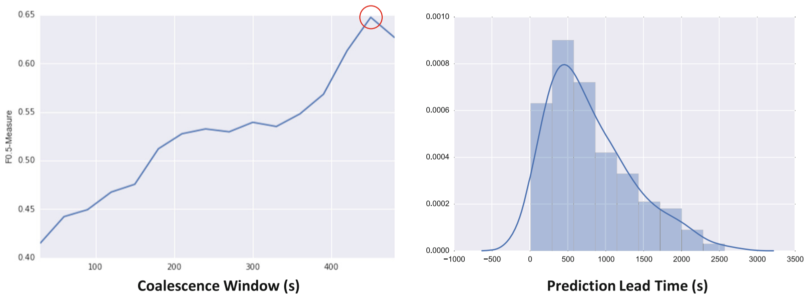

**Fig. 4. Left:** Variation of prediction performance ($F_{0.5}$-Measure) in dependence of the coalescence window. **Right:** Distribution of Monitor's prediction lead time with a kernel density estimator for the optimized model.

Table 2 shows the prediction results for the overall best parametrization ($\Delta t$ = 450 s, $min_{sup}$ = 0.1, $min_{conf}$ = 0.7). By means of comparison, the performance of the state-of-the-art method from `Vilalta et al.` is also presented. We set its time window parameter to be $W$ = 660 s, which optimizes the $F_{0.5}$-Measure.

In total, 89 sequential rules have been mined for creating the predictive monitor evaluated in this work. Without the proposed noisy reduction steps, the maximum achievable prediction performance ($F_{0.5}$-Measure) would fall from 0.65 to 0.62 and the total number of extracted rules would lie by the thousands, which can easily exceed the capabilities of a human operator to identify interesting results. Additionally, our method outperforms the state-of-the-art by both precision and recall, leading to an overall better prediction performance.

[2] Collision occurs when unrelated events are combined into the same group (cluster). The contrary of collision is also denoted “truncation”, referring to related events being separated into different groups when the coalescence window is too small.

**Table 2.** Comparison of prediction performance results between the method proposed by `Vilalta et al.` and the proposed SBFP. Both methods have been parameterized to optimize the $F_{0.5}$-Measure.

| Algorithm | TP | TN | FP | FN | Precision | Recall | FM05 |
|---|---|---|---|---|---|---|---|
| SBFP | 130 | 792 | 65 | 94 | 0.67 | 0.58 | 0.65 |
| Vilaltas' | 124 | 1300 | 80 | 109 | 0.6 | 0.53 | 0.59 |

Number of observations differ due to the different data modeling strategies in both algorithms.

## 6.2 Performance Across Varying Target Events and Datasets

Up to this point, we discussed the main properties and influencing factors of our proposed prediction model for a single target event in a single machine log dataset. It remains to discuss the prediction performance of the approach beyond an unique example. This is the goal of this section.

Naturally, within the scope and capabilities of this work, we can just provide detailed description of the proper parametrization and prediction performance of the proposed approach for selected cases. Nevertheless, in order to provide a broader (but rough) overview of the capabilities of the proposed approach, we perform the following experiment: after the labeling/ranking from domain experts (cf. Sect. 6) we take the remaining 12 target critical events to be predicted, and select the machine log datasets in which the selected target event appears more frequently than a given threshold (in our case in at least 20 distinct sequences). We then run our approach onto those eligible datasets without parameter tuning. Therefore, we can not guarantee that the best solution is found for each run, but we can provide an overview on how the prediction performance roughly varies for a target event across different log datasets from different machines for a given parametrization. The results of this experiment is shown in Fig. 5. For each target event (E1–E12), we show the distribution of the prediction performance when running the algorithm on different datasets in form of Tukey boxplots.

For each run, the prediction performance ($F_{0.5}$-Measure) of the algorithm for the given target event and machine dataset is shown as a point on the boxplots of Fig. 5. As shown in the figure, different events can be predicted to varying degrees. Events $E6$ and $E9$ are for example generally less predictable than most other events with the given parameters. Even for a particular event type, the prediction performance varies significantly according to the machine log dataset used for training and validation of the model. Results for event $E10$ shows a great midspread (IQR) while for other events, like event $E8$, the results dispersion is by orders of magnitude lower. This complies with previous observations of similar work in the literate [4] (cf. Sect. 2). Considering that, the success of the algorithm is contingent on the proper parametrization and existence of sequential pattern of events preceding the target events in the different datasets.

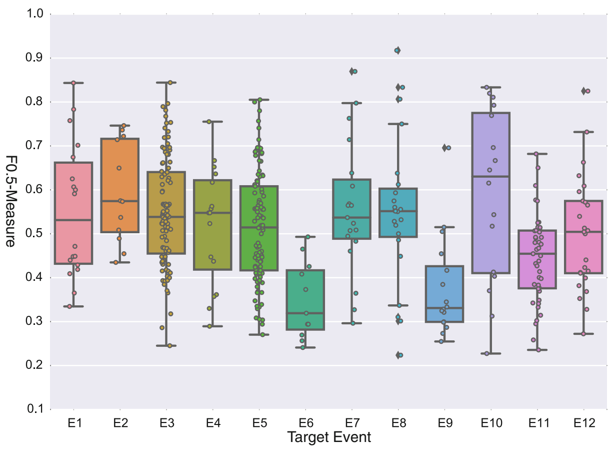

**Fig. 5.** Variation of prediction performance for 12 target event types without parametrization tuning. For each target event, the algorithm was trained and tested individually across several distinct machine log datasets. A point on the graph depicts the prediction performance ($F_{0.5}$-Measure) w.r.t. the given target event for an individual machine log dataset.

## 7 Conclusion

In this work, we analyzed log files generated by industrial computer numerical control (CNC) cutting machines from the company TRUMPF GmbH + Co. KG, with the goal of predicting critical events in machinery log data. We presented a Sequence-Mining based technique for mining sequential patterns of events from log data. Sequence-Mining constitutes an extension to related approaches which so far have just leveraged itemset and association rules mining, which did not consider strict chronological order of events for issuing predictions. We show how our approach with this additional property requires more evidence for issuing positive predictions than methods based just on event sets, leading to a more precise performance outperforming existing techniques.

Furthermore, we presented contributions in how machinery log data can be proper modeled into sequence definitions prior to its usage in sequence mining as well as a information gain based noise reduction technique that guarantees that the detected patterns are related to the target critical event, reducing the result set and increasing prediction's performance.

Our approach is suitable for short-term predictions, achieving prediction lead times of up to 40 min in advance for a specific critical event.

Together with domain experts, we extensively evaluated our approach and its parametrization using real data from over 150 machines containing millions of log entries and discuss the benefit of such techniques for the industry.

## References

1. Raman, S., De Silva, C.W.: Sensor-fault tolerant condition monitoring of an industrial machine. Int. J. Inf. Acquis. **9**(01), 1350001 (2013)
2. Salfner, F., Tschirpke, S.: Error log processing for accurate failure prediction. In: WASL (2008)
3. Salfner, F., Lenk, M., Malek, M.: A survey of online failure prediction methods. ACM Comput. Surv. **42**(3), 1–42 (2010)
4. Vilalta, R., Ma, S.: Predicting rare events in temporal domains. In: Proceedings of the 2002 IEEE International Conference on Data Mining, 2002. ICDM 2003, pp. 474–481. IEEE (2002)
5. Tsao, M.M., Siewiorek, D.P.: Trend analysis on system error files. In: Proceedings of the 13th International Symposium on Fault-Tolerant Computing, Milano, vol. 116119 (1983)
6. Hansen, J.P., Siewiorek, D.P.: Models for time coalescence in event logs. In: Twenty-Second International Symposium on Fault-Tolerant Computing, FTCS-22. Digest of Papers, pp. 221–227. IEEE (1992)
7. Lal, R., Choi, G.: Error and failure analysis of a unix server. In: Proceedings of the Third IEEE International on High-Assurance Systems Engineering Symposium, pp. 232–239. IEEE (1998)
8. Zaki, M.: SPADE: an efficient algorithm for mining frequent sequences. Mach. Learn. **42**(1–2), 31–60 (2001)
9. Zhao, Q., Bhowmick, S.S.: Sequential pattern mining: a survey, ITechnical report CAIS Nayang Technological University Singapore, pp. 1–26 (2003)
10. Slimani, T., Lazzez, A.: Sequential mining: patterns and algorithms analysis. arXiv preprint arXiv:1311.0350 (2013)
11. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**, 281–305 (2012)