Explainable Failure Predictions with RNN Classifiers based on Time Series Data

Ioana Giurgiu
IBM Research - Zurich
igi@zurich.ibm.com

Anika Schumann
IBM Research - Zurich
ikh@zurich.ibm.com

Abstract

Given key performance indicators collected with fine granularity as time series, our aim is to predict and explain failures in storage environments. Although explainable predictive modeling based on spiky telemetry data is key in many domains, current approaches cannot tackle this problem. Deep learning methods suitable for sequence modeling and learning temporal dependencies, such as RNNs, are effective, but opaque from an explainability perspective. Our approach first extracts the anomalous spikes from time series as events and then builds an RNN classifier with attention mechanisms to embed the irregularity and frequency of these events. A preliminary evaluation on real world storage environments shows that our approach can predict failures within a 3-day prediction window with comparable accuracy as traditional RNN-based classifiers. At the same time it can explain the predictions by returning the key anomalous events which led to those failure predictions.

Introduction

Explainable predictive modeling based on telemetry data is key in many domains, from healthcare to IT and industries, and it is particularly challenging when concerned with critical incidents. In IT environments, these incidents represent failures of devices or components and are typically very rare ($<$ 2-3% of all incidents). Even being able to predict a small fraction of them significantly increases availability, generates savings and avoids labor costs, as opposed to taking the current approach where maintenance and repair operations are performed reactively, after the critical incidents occur.

As a result, up until recently the primary concern from an AI perspective was to build models that can predict such failures ahead of time as accurately as possible. This has led to a transition from traditional ML approaches, such as random forests and gradient boosted machines, to deep-learning methods because of their superior performance. In particular, RNNs are effective for use cases that benefit from sequence modeling and learning temporal dependencies. However, in the last few years it has become clear that such predictive models are only applicable in practice if they provide some degree of usable intelligence (**?**; **?**). That implies they need to be able to learn from prior data, generalize well and extract the explanatory factors of the data (**?**). Therefore, the current objective is to build accurate AI models that at the same time provide explanations intelligible to domain experts. Numerous approaches have been proposed (**?**; **?**; **?**; **?**; **?**; **?**), either providing post-hoc explainability (i.e., agnostic to the underlying black-box model) or ante-hoc explainability (i.e., incorporating explanations in the black-box model itself). What they have in common is their application primarily on images and text. This is most probably due to two reasons: 1) explanations around text or images are easier to comprehend by humans (e.g., an explanation highlighting a guitar in an image makes sense for a classifier that is supposed to determine the presence or absence of a guitar, and provides confidence in the underlying classifier), and 2) there is no time component involved.

In this paper, we are concerned with explaining predicted failures in storage environments, based on key performance indicators (KPIs) collected with fine granularity as time series. Given the temporal component, using RNNs as the underlying classifier is a natural choice. While using post-hoc explainable models is attractive, since it does not require changes to the black-box model itself, to the best of our knowledge there are no approaches built specifically for temporal data. We show that applying LIME (**?**) to our time series KPIs floods the human expert with a vast number of imprecise explanations in the shape of highlighted portions of the series.

Therefore, we consider incorporating explainability directly into the classifier. Based on the observations that our KPIs are spiky rather than exhibiting increasing or decreasing trends (Fig. 1), and that the progression and accumulation of spikes over time can lead to critical incidents, we are inspired by the medical domain, where RNN-based approaches for diagnosis have modeled the temporal progression of an illness as event series with decaying factors (**?**). Therefore, we transform our time series into series of clustered anomalous events, where these events are defined as $KPI^t_{val} > threshold$ (KPI^t_{val} is the value of a KPI at time t). Such clusters exist, because anomalous events have a tendency to co-occur within well-defined time intervals. Moreover, they appear at random points in time and incur a bursty behavior – long periods with no or few events, followed by shorter periods with multiple events. Traditional RNNs are oblivious to the time interval between two clus-

ters. Recent efforts use the information about these time intervals to compute the hidden states of the recurrent unit (**?**; **?**; **?**), but do not consider that the health status of a device and the time between events are correlated. Intuitively, the more frequent and recent the anomalous events, the more impact they will have on future critical incidents.

In short, our approach follows multiple steps: 1) clusters (windows) of anomalous events are detected optimally via Ckmeans.1d.dp (**?**); 2) unique anomalous events are embedded in a continuous vector space; 3) for each event in a cluster, context information is aggregated using the attention mechanism proposed in (**?**); 4) for each event, we build a temporal progression function that quantifies how much of an impact the event has on the prediction objective, depending on its type and when it occurred; 5) using the context information and the progression function, each window is represented as a weighted sum of embeddings of its events; and 6) the window representations are used in an LSTM network to predict failures within a predefined interval. We note that while multiple anomalous events of the same type can occur in a window (Fig. 3), no progression is assumed within the same window.

We conducted a preliminary evaluation on KPIs collected from 130+ storage environments over 14 days in May and June 2018, respectively, with 5 minute granularity. Based on threshold rules defined by domain experts, we extract 266081 anomalous events and use them to predict storage failures within 3 days after each 14-day interval. Initial results show that our approach can achieve comparable accuracy with traditional RNN-based models, while at the same time it provides useful insights into its prediction decisions.

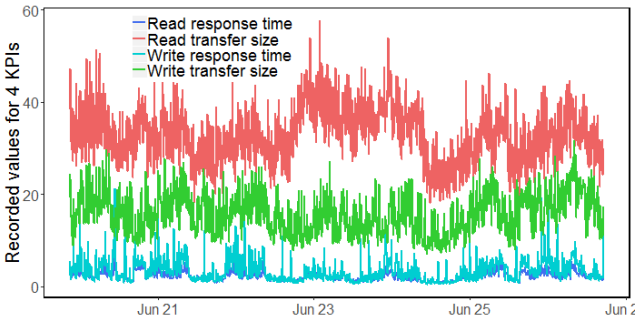

Figure 1: Time series of Read/Write response time and Read/Write transfer size collected over 10 days for a device.

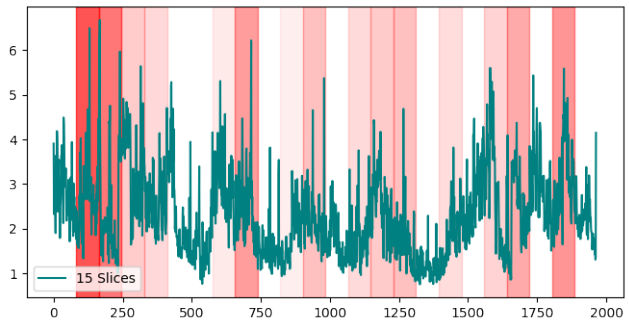

Figure 2: Highlighted slices for Read response time series over 10 days, extracted with LIME (15 slices, 82 points per slice). Darker red indicates higher contribution to prediction.

Motivation

Our approach is driven by the characteristics of our data.

Spiky nature of KPIs – Although there is an expectation that KPIs should exhibit either an increasing or decreasing trend prior to a critical event, it is very common for them to actually be spiky in nature (with their values at times exceeding pre-defined thresholds), but maintain relatively constant means through time. In Fig. 1, we show such an example, namely the time series collected for 4 KPIs over a period of 10 days for one storage device that fails 2 days later. To confirm the spikeness of our data beyond visual inspection, we apply changepoint detection via causal Bayesian inference (**?**). The underlying premise of changepoint detection is that when a KPI has impact over a future critical incident, there should be a significant shift at a timestamp before the incident occurs. Even more important, this shift should be permanent. Shortly, for a time series $S_i = (s_1, s_2, \cdots, s_p)$ of KPI i, if there exists a timestamp $t < p$ when a significant change in S_i occurs (e.g., values start increasing), then S_i potentially has impact over a future critical event. To verify whether the change is permanent, we check if the difference between the time series of the KPI and the corresponding time series in the absence of the change at time t (i.e., synthetic series) is significant. To generate the synthetic series, we compute the posterior distribution $p(s_{t+1:p} \mid s_{1:t}, x_{1:p})$, where $s_{1:t}$ is the series in the pre-change period and $x_{1:p}$ is the control time series generated from healthy devices. Finally, the sensor metric has impact if the probability distributions of the actual time series after the detected changepoint and the synthetic series are significantly different (i.e., $p < 0.05$). With all our KPIs, the probability distributions are not different ($p \gg 0.05$).

LIME for time series – Even considering the spiky nature of the KPIs, it is still attractive to apply agnostic models, such as LIME (**?**), to understand how various KPIs contribute to a prediction. Therefore, we use a binary classifier to tackle the prediction problem introduced previously – taking as input KPI time series collected over 14 days to predict whether a failure will occur within the next 3 days. Then, we apply a parametric time series implementation of LIME (**?**) to understand which KPIs and portions of their corresponding series contribute to individual predictions. Fig. 2 highlights the portions for read response time from Fig. 1 that contributed to the failure incident recorded 2 days later. We note the following limitations: 1) the quality of the explanation highly depends on the number of slices given as an input parameter and it involves significant trial and error; 2) the highlighted portions in the series have a fixed length (i.e., size of the slice); 3) the lower the number of slices, the less discriminative a series portion becomes; 4) more slices result in a vast number of explanations per KPI that are difficult to follow even by a domain expert; 5) there is no tempo-

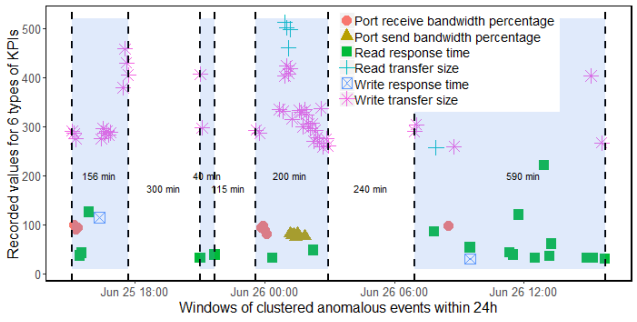

Figure 3: Anomalous events clustered in 4 highlighted time windows of variable length during 24h for a storage device.

ral consideration – in the example, the highest contribution is attributed to the highest jumps in the read response time, which happen to occur in the first day of the series (12 days before the failure). This is completely opposite to how a system behaves, namely the more recent the spikes, the more impact they will have on future critical incidents.

Events co-occurence – Driven by the spiky behavior of the KPIs and the limitations of applying LIME, we consider using the spikes as anomalous events rather than the entire time series. Spikes can be captured either by basic thresholding (e.g., read response time $\geq$ 200 ms/op) or by detecting them as anomalies using AI techniques. While numerous anomaly detection algorithms have been proposed in recent years (**?**; **?**; **?**), the unsupervized phase is always followed by a semi-supervized phase, namely the anomaly validation. Typically validating anomalies has to be done with a human in the loop, even if the manual validation effort would only involve a small sample of the detected anomalies. For the purpose of this paper, we revert to using expert-defined thresholds to capture the spikes. As a result, we uncover an interesting pattern: these spikes have a tendency to co-occur within relatively compact time windows, separated by windows where no spikes appear. Fig. 3 highlights 4 windows of clustered spikes captured within 24h for a storage device. All windows are of variable length and contain an arbitrary number of spikes recorded pertaining to 6 KPIs. For all other KPIs, no spikes exceeding pre-defined thresholds were recorded. The challenge is to detect these windows. We detail our method next.

Method

In this section, we describe our proposed prediction model. First, we introduce the problem setup. Second, we describe how we detect the windows of anomalous events. Then, we detail our RNN-based approach, and finally we explain how we provide explanations based on analyzing weights associated to anomalous events.

Problem Setup

Our dataset is a collection D of d storage devices. For each device, a set KPI of M metrics are collected as time series at regular time intervals over a period t. Across all M time series pertaining to a device D_i, we collect the set of l anomalous events $E_{D_i} = \{e_{1D_i}, ..., e_{lD_i}\}$. Each anomalous event refers to one KPI and is registered when a predefined threshold is exceeded, such that $KPI_j > th_{KPI_j}$. Next, we cluster the events for each device D_i into a set of windows $W_{D_i} = \{W_{1D_i}, ..., W_{pD_i}\}$, where $p < l$, ordered chronologically. Window W_{rD_i} is a 2-dimensional vector $W_{rD_i} = \{\beta_{rD_i}, t_{rD_i}\}$ of unordered anomalous events $\beta_{rD_i} = \{e_{1D_i}, ...e_{|\beta_{rD_i}|D_i}\}$ and corresponding timestamps. We denote with $\mathbf{E}$ the vocabulary of events and its cardinality with $|\mathbf{E}|$.

Detecting Windows of Anomalous Events

In the first stage, we detect all windows of anomalous events for a device, $W = \{W_1, ..., W_p\}$, $p < l$, within the time interval $[0, t]$. Since our problem is one-dimensional (1-D), we use an optimal k-means clustering algorithm with dynamic programming, called Ckmeans.1d.dp (**?**). We essentially assign anomalous events of the input 1-D array E into k clusters so that the sum of squares of inner-cluster distances from each event to its corresponding cluster mean is minimized. The beauty of Ckmeans.1d.dp is that it can estimate k optimally on its own, using Bayesian information criterion. An example of detected windows is shown in Fig. 3. As seen, all windows are of arbitrary length and clearly separated one from the other.

Representing Windows of Anomalous Events

Next, the objective is to predict future critical incidents – particularly failures – for a device D_i, within a window $[t, t + T]$, by using the corresponding windows of anomalous events, $W_1, ..., W_p$. An essential aspect is how we represent each window W_r, $r < p$, as a vector. A simple approach is to embed every event in $\mathbf{E}$ in a continuous vector space, such that the s-dimensional vector $v_e \in R^s$. Then, the vector representation of a window W_r, w_r, containing N events $\beta_r = \{e_1, ..., e_N\}$ is the sum of embeddings of events occuring within the window, namely

$$w_r = \sum_{n=1}^{N} x_{e_n} v_{e_n} \tag{1}$$

where x_{e_n} represents how many times event e_n occurred within window W_r. The problem with this approach is that the impact of an event is proportional to its frequency. If all events occur only once, they contribute equally to the window representation. Instead, the window representation should also depend on when an event occurs and what it is. Given that we treat each window as a set of unordered events (since we cannot assume causality in the events sequence), we consider using attention mechanisms (**?**), which have been successfully applied in language translation. We define the context vector cv_n for an event e_n as

$$cv_{e_n} = \sum_{x=1}^{N} \alpha_{nx} v_{e_x} \tag{2}$$

where the attention value is $\alpha_{n1}, ..., \alpha_{nN} = softmax([\frac{q_n k_1}{\sqrt{a}}, ..., \frac{q_n k_N}{\sqrt{a}}])$, as defined in (**?**). q_n is the

query vector for e_n, k_n is the key vector for the same event and a is their dimension.

Now that the context vector for an event e_n in a window is defined, we need to quantify each event's contribution to the prediction representation in $[t, t+T]$. As already stated, the contribution of each event depends on when the event occurred. The further in the past, the smaller the contribution. Therefore, we define the contribution as follows:

$$I(c_{e_n}, \Delta) = S(\theta_{e_n} - \sigma_{e_n}\Delta) \in [0, 1] \tag{3}$$

where S is the sigmod function, θ_{e_n} is the initial contribution of e_n to the prediction, σ_{e_n} is the progression of this contribution function of time and Δ is the time elapsed from window W_j to the end of prediction window, namely $\Delta = t + T - \tau_{W_j}$.

Finally, we rewrite the vector representation of a window w_r as follows:

$$w_r = \sum_{n=1}^{N} x_{e_n} I(c_{e_n}, \Delta) c v_{e_n} \tag{4}$$

Explanaible Predictions with LSTMs

We use the window representations, $w_1, ..., w_p$, as inputs to the LSTM to predict a label $y \in \{0,1\}$, which represents whether a critical event will occur or not in the $[t, t+T]$ interval. Note that our problem reduces to binary classification and the predicted event does not belong to the vocabulary $\mathbf{E}$, such that:

$$\hat{y} = \sigma(W h_p + b) \tag{5}$$

where h_p is the hidden state output at step p of the LSTM, W is the weight matrix, b is the bias vector of the output function and $\hat{y}$ is the predicted label probability distribution.

We explain predictions by quantifying how much each anomalous event within a window W_r contributed to the decision. For each embedding v_{e_n} of event e_n, there are associated weights defining that contribution. More specifically,

$$\begin{aligned} w_r &= \sum_{n=1}^{N} x_{e_n} I(c_{e_n}, \Delta) \sum_{x=1}^{N} \alpha_{nx} v_{e_x} \\ &= \sum_{x=1}^{N} (\sum_{n=1}^{N} x_{e_n} I(c_{e_n}, \Delta) \alpha_{nx} v_{e_x}) \end{aligned} \tag{6}$$

where $\sum_{n=1}^{N} I(c_{e_n}, \Delta)\alpha_{nx}$ is the contribution of each event embedding and x_{e_n} is the frequency of occurrence of each event.

Preliminary Evaluation

Data and Setup

KPIs and Events – We collect KPIs as time series for 130+ storage environments over 14 days in May and June 2018, respectively, with 5 minute granularity. As storage environments are typically complex, the KPIs are collected across their entire hierarchy (e.g., nodes, ports, volumes, RAID arrays, disks). More specifically, each storage device is mapped to multiple nodes (2-4 on average) connected to tens or hundreds of hosts, through a varying number of ports (a power of 2 in the range [4,32], each with 8 or 16 GB/s speed), depending on the fabric architecture. For each environment, we collect all the anomalous events registered for both 14 days intervals, according to the threshold rules defined by domain experts. In total, there are 266081 such events, distributed across 15 KPI rules, as shown in Table 1.

KPI	Threshold	#
Disk utilization	50 %	4448
Invalid transmission word rate	0.7 cnt/s	4331
Peak backend write resp. time	10s	238
Port receive bandwidth	75%	4314
Port send bandwidth	75%	332
Port send delay I/O	20%	55170
Port to local node send queue time	0.5ms/op	720
Port to local node send resp. time	0.75ms/op	20666
Read response time	30ms/op	49010
Read transfer size	64KB/op	79100
System CPU core utilization	70%	706
Write-cache delay	3%	112
Write response time	30ms/op	52424
Write transfer size	256KB/op	44473
Zero buffer credit	20%	37

Table 1: Anomalous events distributed across 15 KPI rules.

Critical incidents – Additionally, we collect all the incidents registered within 3 days of the end of each time interval, as these represent our prediction goal. These incidents capture a variety of problems, such as "running out of space", "device health is below standard", "drive has excessive errors interfering with the hardware", "software level is below recommended version", "battery is at end of life" or "drive not responding to commands". They are classified based on severity in *informational* (87%), *warning* (9%) and *error* (4%). We focus on *error* incidents and keep only those regarding devices or drives and containing the phrase "the device or drive is likely to fail soon". From hundreds of incidents collected, these are less than 2% and refer to 3% of the devices. This implies a 1:32 ratio between the minority (i.e., failure) and majority classes.

Design and Metrics – We compare our model with two other binary classifiers, namely random forest (RF) and traditional LSTMs. The RF model, implemented in R, is optimized to use the optimal number of trees, oversampling ratio of the minority class and the number of samples at leaf nodes. In addition, we assign double weight to the minority class to penalize miss-classified failures more. The LSTM model is a traditional network, where the sum of the event embeddings are used as window representations (Equation 1) and fed as inputs. It is implemented in Keras and uses a batch size of 50, a sequence length of 15 and a learning rate of 0.1. We use the same LSTM parameters for our model. For both datasets, we randomly sample 80% for training and the rest of 20% for testing and validation. We report precision and F1-score values for the minority class, as well as the balanced accuracy across both classes.

Windows Size

We analyze the number of windows obtained when clustering anomalous events per each device, as well as their size (i.e., the number of events within a window). The distributions are shown in Fig. 4. Typically, the number of clusters varies from 1 to 10 (rarely to 15) and the size can go up to 1500 events. Single clusters per device are very rare and appear only when there are few events recorded in a short time span. These are specific to devices that have fewer ports and nodes, and are connected to less hosts. For devices that have very complex environments (e.g., hundreds of servers connected and hundreds or thousands of volumes), we record thousands of anomalous events. In this case, both the number of the clusters and their size increases, which explains the shapes of the distributions. Fig. 5 shows the 3 clusters obtained with the Ckmeans.1d.dp algorithm for a device with 90 anomalous events recorded over 24h. The clusters are of sizes 22, 47 and 21, respectively.

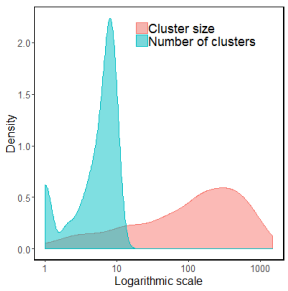

Figure 4: Distribution of cluster sizes and number of clusters per device.

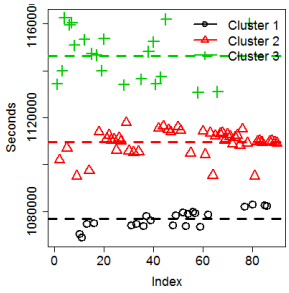

Figure 5: Example of 3 clusters for a device with 90 anomalous events.

Prediction Accuracy

Table 2 shows the accuracy and F1-score values for the minority class, as well as the balanced accuracy across both minority and majority classes, obtained with the RF, LSTM and LSTM with attention models. On the one hand, RF is outpeformed by both neural network models, which is to be expected considering that RFs are not built to specifically deal with temporal data. On the other hand, the gains in accuracy obtained with both LSTM models are relatively modest. We attribute this to: 1) the challenge of the prediction problem, since the data set is highly imbalanced (1:32 ratio between minority and majority classes); 2) the lack of hyperparameter optimization for both LSTM implementations.

However, our goal was to build a model that does not sacrifice accuracy in favor of explainability. As seen, the LSTM with attention approach is slightly better than the traditional LSTM, while at the same time being able to provide prediction interpretability, as shown next.

Explainability

The model provides explainability by associating weights with each anomalous event as a function of its frequency within each window and the contribution of its embedding

Model	Prec.1	F1.1	bACC
RF	0.24	0.19	0.55
LSTM	0.28	0.23	0.59
LSTM with attention	**0.29**	**0.24**	**0.60**

Table 2: Precision and F1-score on the minority class, and balanced accuracy for RF, LSTM and LSTM with attention.

(Equation 6). We show how our approach works for 3 storage devices for which our predictions indeed occur within the prediction window. The first device is predicted to fail with a probability of 0.87, while the second and third are predicted to fail with a probability of 0.23 and 0.31, respectively (i.e., 0.77 and 0.69 probabilities not to fail).

Prediction: Fail – For this device, there are 69 anomalous events recorded during 14 days in June. These events are clustered in 15 windows as shown in Table 3. For each window, we show the number of events distributed by type and the weights associated. As seen, most assigned weights are either 0 or < 0.01 as they do not contribute to the prediction. Such sparsity in the weights distribution is useful for domain experts interpreting a prediction, since the model focuses on events with non-negligible predictive power. For instance, our approach strongly prefers the events occuring in the last 3 windows, and more specifically, those related to write response times thresholds being exceeded (i.e., write response time and peak backend write response time). While higher than expected write response times are already signaling performance decline, together with peak backend write response times (i.e., the longest time for a back-end storage resource to respond to a write operation by a node) they would indicate an impeding critical incident, such as a failure. Our model learns such associations between anomalous events. In addition, it learns that read response time events contribute significantly less to the failure (i.e., 0.04 in window 15 as opposed to 0.4 and 0.21 for peak backend write response time and write response time, respectively) and their weights decrease the more distant they are to the prediction window.

Prediction: No fail / Example 1 – Table 4 shows the weights associated to 22 anomalous events clustered in 7 windows for a device that remains healthy throughout the 3-day prediction window. We note the following. First, the model learns that events such as disk utilization threshold exceeded do not lead to critical incidents (i.e., < 0.01), since they do not accumulate over consecutive windows. This is due to the fact that cleaning jobs are run periodically to remove temporary files, therefore decreasing disk utilization. Second, the temporal progression of events still matters, since events weigh more if they occur closer to the start of the prediction window. Third, certain types of events (e.g., read response time, read transfer size) generally have more predictive power than others (e.g., disk utilization).

Prediction: No fail / Example 2 – Finally, we show that the time when a predictive anomalous event occurs impacts the actual prediction. In the first example, the peak backend write response time coupled with the write response time in day 14 had the largest impact on the fail probability. In

Window	Start timestamp	Duration	# Events	Event	Frequency	Contribution
1	Day 1 22:58	115 min	11	Read response time Read transfer size Write transfer size	1 5 5	0.00 0.00 0.00
2	Day 2 6:04	105 min	1	Read response time	1	0.00
3	Day 2 20:19	390 min	8	Read response time Read transfer size Write transfer size	1 3 4	<0.01 <0.01 0.04
4	Day 3 19:59	195 min	6	Read transfer size Write transfer size	3 3	<0.01 0.06
5	Day 4 23:00	70 min	3	Read transfer size Write transfer size	1 2	<0.01 0.06
6	Day 5 6:15	120 min	2	Read response time	2	0.015
7	Day 5 22:55	20min	2	Read response time	2	0.02
8	Day 6 22:56	20 min	1	Read transfer size	1	<0.01
9	Day 7 23:01	15 min	2	Read transfer size	2	0.01
10	Day 8 6:02	125 min	3	Disk utilization	3	0.00
11	Day 8 22:57	20 min	9	Read transfer size Write transfer size	5 4	0.05 0.16
12	Day 9 23:12	65 min	3	Read response time	3	0.06
13	Day 11 20:28	205 min	4	Write response time	4	0.18
14	Day 13 4:08	35 min	6	Read response time Write response time	4 2	0.1 0.34
15	Day 14 22:59	15 min	8	Read response time Peak backend write response time Write response time	3 2 3	0.12 0.8 0.63

Table 3: Weights associated with 69 anomalous events clustered in 15 windows for a storage device predicted to fail.

Table 5 we highlight only the occurence of peak backend write response time events for a device that does not fail. Even though these events occur multiple times, they are too distant from the start of the prediction window, thus their weights are low (i.e., 0.025 and 0.04). Additionally, no write response time events occur in the same windows. Therefore, our model is sensitive to events frequency, their exact timestamps and their co-occurence with correlated event types.

Limitations

As this work is only in its initial phase, it is by no means complete. We identify a few limitations to be addressed next.

Data set size – Given the low failure rate in our use case, ideally we should use series of KPIs collected over months or even years to increase the size of the minority class and improve prediction accuracy. For this paper, we wanted to show a proof of concept of how we can provide explanations with LSTMs, while we are continuing to collect data.

Multivariate anomalous events – So far, we focused on single KPI thresholds to identify anomalous events. We plan to use anomaly detection algorithms to explore multivariate time series and identify complex anomalous patterns instead. First, we expect to uncover seasonal patterns of behavior that are not obvious to an expert. Second, we believe model performance will increase and the sizes of clusters will reduce, making it easier for a domain expert to use the explanations provided. However, our primary goal was to show how we can marry LSTMs with built-in explainability to provide interpretable predictions.

Advanced DNN models – We are aware that prediction accuracy can be improved by using more advanced DNN models, such as bidirectional LSTMs, RNNs with gated recurrent units (**?**) or even combinations of LSTMs and CNNs (**?**). In the latter, fully convolutional blocks and LSTM units are run in parallel and their outputs are passed to a softmax classification layer. We plan to try out more advanced models going forward, while keeping the attention layer in place.

Related Work

While there exists a lot of work around explainable models for images and text, little attention has been been given to explaining models based on temporal data, namely time or event series. On the one hand, post-hoc approaches aim at explaining a model's prediction after the event, which means they should be agnostic and applicable on any type of data. On the other hand, ante-hoc methods incorporate explainability directly into the black-box model, which implies they are tailored to the underlying model and data.

Post-hoc approaches aim to provide local explanations for specific decisions, rather than attempting to explain the whole system behavior. One of the most representative examples for classification in recent years is LIME (**?**). The approach is simple: generate an explanation by approximating the underlying model by an interpretable one (e.g., a linear model with a only a few non-zero coefficients), learned on perturbations of the original instance. Typical perturbations can be removing words or hiding parts of an image. A similar model-agnostic approach is BETA (**?**), which optimizes for fidelity to the black-box model and interpretability

Window	Start timestamp	Duration	# Events	Event	Frequency	Contribution
1	Day 1 10:07	65 min	1	Disk utilization	1	0.00
2	Day 3 10:02	395 min	9	Disk utilization Read transfer size	5 4	0.00 0.00
3	Day 4 2:07	195 min	2	Read transfer size	2	<0.01
4	Day 6 8:37	15 min	2	Read response time	2	<0.01
5	Day 10 15:07	25 min	1	Read response time	1	0.04
6	Day 11 18:22	65 min	2	Read transfer size	2	0.05
7	Day 13 2:47	135 min	5	Read response time Disk utilization	2 3	0.04 0.02

Table 4: Weights associated with 22 anomalous events clustered in 7 windows for a storage device predicted not to fail.

Window	Start timestamp	Duration	# Events	Event	Frequency	Contribution
1	Day 2 15:17	35 min	5	Peak backend write response time Read response time	2 3	0.05 0.00
2	Day 5 12:02	105 min	2	Peak backend write response time	2	0.06

Table 5: Highlighted weights associated with 5 peak backend write response time anomalous events clustered in 2 windows for a storage device predicted not to fail.

of the explanation. (**?**) focuses on pixel-wise decomposition of nonlinear classifiers, which allows to visualize contributions of single pixels to predictions for kernel-based classifiers. (**?**) extracts explanations from latent factor recommendation systems by training association rules on the output of a matrix factorization black-box model. All approaches have been applied on text and images, but are not built to take into consideration temporal progressions in time or event series.

Ante-hoc approaches are interpretable by design (**?**). Typical examples include decision trees, decision sets (**?**; **?**), fuzzy inference models (**?**) or additive models (**?**). However, none of these fit temporal data well. The vast majority of explainable models for time series target their classification. (**?**) propose grammar-based decision trees to classify heterogeneous time series. (**?**; **?**) extract interpretable features from series, expressed as local shapelets, while (**?**) learn such shapelets via stochastic gradient learning and use them for early classification. In (**?**), the authors propose reversible and irreversible explainable tweaking, where given a time series and an opaque classifier, the objective is to find the minimum number of changes to the time series such that the classifier changes its decision.

Closest to our problem is the method proposed in (**?**). There, the objective is to predict a future neural event based on a sequence of previously occurred events. Current approaches are mostly concerned with time-independent sequences, in which the actual time span between events is irrelevant and the difference between events is the difference between their order positions in the sequence. The authors extract and use the information provided by the time span between events in an RNN-based model to achieve some accuracy gain over baseline models. We also opt for an RNN architecture, but we additionally incorporate attention mechanisms (**?**) into the network to quantify how much an anomalous event contributed to a predicted critical incident.

Conclusions

Predictive modeling based on temporal data is key in many domains, from healthcare to IT and industries, particularly when it is concerned with critical incidents, such as failures. Providing explanations for these predictions is crucial, as it enables experts to gain trust in AI-powered models and take into consideration their outputs in the decision process. State of the art explainable models mostly focus on images and text and are not easily applicable to time or event series. We propose a deep learning approach that takes into consideration the irregularity and frequency of anomalous events extracted from time series and uses attention mechanisms to aggregate context information of these events in order to quantify how much information from each event flows into the network. A preliminary evaluation on 266081 events collected from real world storage environments shows that our approach is comparable in accuracy with traditional LSTMs, while at the same time being able to quantify the contribution of each past event recorded to a failure prediction.

Quis asperiores assumenda in non at aspernatur doloremque, eum at placeat adipisci, atque alias veritatis nostrum iure assumenda excepturi doloribus, non iusto maxime hic distinctio omnis nisi ipsa, deleniti sit ut recusandae obcaecati doloremque expedita?A repellendus hic consectetur natus suscipit est, dolorem quod debitis temporibus doloribus atque iste dolore nobis at, mollitia enim possimus porro dolores saepe quis nemo fugit sapiente aperiam incidunt.Assumenda voluptates temporibus nisi voluptatum et quis aspernatur unde, ipsa dolor fugit, dolores libero et vel tempora autem magnam perspiciatis labore hic, perferendis iste est numquam nostrum modi possimus recusandae maxime tempora hic.Deserunt cum quasi eos atque facilis minus, maxime eius sint quibusdam quam nulla deleniti soluta, pariatur laborum blanditiis facere rem dicta officia aut aliquid soluta, explicabo excepturi minima vitae fugiat enim voluptate.Reiciendis maiores quia veritatis ratione itaque quisquam perspiciatis consequuntur repellat, explicabo ipsum itaque tempore consequuntur accusamus illo distinctio mollitia expedita fuga optio.Nesciunt nihil