Mining Multivariate Discrete Event Sequences for Knowledge Discovery and Anomaly Detection

Bin Nie
William and Mary
Williamsburg, VA
bnie@cs.wm.edu

Jianwu Xu
NEC Laboratories America
Princeton, NJ
jianwu@nec-labs.com

Jacob Alter
William and Mary
Williamsburg, VA
jralter@email.wm.edu

Haifeng Chen
NEC Laboratories America
Princeton, NJ
haifeng@nec-labs.com

Evgenia Smirni
William and Mary
Williamsburg, VA
esmirni@cs.wm.edu

***Abstract*—Modern physical systems deploy large numbers of sensors to record at different time-stamps the status of different systems components via measurements such as temperature, pressure, speed, but also the component's categorical state. Depending on the measurement values, there are two kinds of sequences: continuous and discrete. For continuous sequences, there is a host of state-of-the-art algorithms for anomaly detection based on time-series analysis, but there is a lack of effective methodologies that are tailored specifically to discrete event sequences.**

This paper proposes an analytics framework for discrete event sequences for knowledge discovery and anomaly detection. During the training phase, the framework extracts pairwise relationships among discrete event sequences using a neural machine translation model by viewing each discrete event sequence as a "natural language". The relationship between sequences is quantified by how well one discrete event sequence is "translated" into another sequence. These pairwise relationships among sequences are aggregated into a *multivariate relationship graph* that clusters the structural knowledge of the underlying system and essentially discovers the hidden relationships among discrete sequences. This graph quantifies system behavior during normal operation. During testing, if one or more pairwise relationships are violated, an anomaly is detected. The proposed framework is evaluated on two real-world datasets: a proprietary dataset collected from a physical plant where it is shown to be effective in extracting sensor pairwise relationships for knowledge discovery and anomaly detection, and a public hard disk drive dataset where its ability to effectively predict upcoming disk failures is illustrated.

***Index Terms*—anomaly detection, categorical event sequences, discrete event sequences, rare events, unsupervised learning, physical plant failures, disk failures**

I. Introduction

Today's information technology systems consist of many heterogeneous components working concurrently. Similarly, a typical industrial plant contains heat-generating units, turbine/generator units, condensers, and pump systems. Hundreds to thousands of sensors of various types may be deployed in these components to monitor a host of system attributes. The collected data are analyzed to obtain system status; this may include being fed into log analytics engines to learn about system structure, detect anomalous behavior, aid system administration and maintenance, and/or diagnose system failures [6], [8], [9], [18], [21], [28] In a physical plant, the collected sensor data may consist of continuous measurements (e.g., temperature, pressure, utilization) and/or discrete ones (e.g., system states such as ON/OFF or command execution orders). Discrete data typically have a categorical format. For example, a software controller may record event information in the form of time stamps that signify the start and completion of a job.

The challenge here is to use measurements to understand the joint behavior of different system components [7], [29], [36], [39]. One way to model joint behavior is to model *pairwise* relationships among system components using regression [17]. This has the benefit of using far fewer parameters than modeling a full joint distribution of the data and can lead to more readily interpretable results. However, the fact that different sensors record system states in the form of *continuous time series* (i.e., consisting of numerical variables) or *discrete event time sequences* (i.e., consisting of categorical variables) introduces additional difficulties as regression models are not readily applicable to discrete sequences. Categorical variables cannot be meaningfully assigned to numeric values. For example, if a sensor has three levels—low, medium, and high—these could easily be assigned the values $\{0, 1, 2\}$ or equivalently $\{1, 10, 100\}$. Furthermore, many regression models are fit based on the assumption of Gaussian error distributions, which is unhelpful in the case of discrete data.

In practice, a large percentage of signals collected in complex systems are in the format of categorical variables [21]. For example, in a typical physical plant, the percentage of sensors that produce discrete event sequences can be as high as 90%. Operation management technologies designed for continuous sequences have to either discard the discrete event sequences or rely on extensive feature engineering efforts to "translate" discrete sequences in continuous format, which requires domain-specific knowledge [46].

An additional challenge is that anomalies are typically very rare. For example, for the proprietary dataset used in this paper, there are only two anomalies in a month. Similarly, for the two datasets in [45], the number of anomalies is less than three per month. Such datasets are extremely imbalanced [6], [29], [30] and the use of supervised learning on those is infeasible as supervised learning would require years of system measurements to collect enough anomalies in order to achieve good performance for the minority class [24], [44], [45]. Unsupervised learning algorithms such as K-Means [10], [43] and one-class SVM [35] can be used instead, as they build models based entirely on samples from normal system operation periods and detect outliers that fall out of the learned distribution [4], [35], [43]. Unfortunately, existing unsupervised algorithms require continuous sequences as input.

In order to resolve the aforementioned challenges, this paper presents a novel framework for knowledge discovery and anomaly detection for multivariate discrete event sequences. In the remaining of this paper and for ease of presentation, we assume that each discrete event sequence is generated by a sensor. Knowledge discovery is achieved by mining the interdependence relationship of sequences during *normal* system operating times via the creation of pairwise relationships among sensors generating discrete event sequences. The pairwise sensor relationships are organized in the form of a *multivariate relationship graph*. This directed graph can be used to cluster sensors into connected components in subgraphs that capture physical or functional relationships among sensors. Since the multivariate relationship graph reflects normal system operation, when an anomaly occurs, the relationships among sensor pairs may break down. These broken relationships serve as signs of abnormal behaviors.

Core to the proposed solution is the identification of an effective metric to quantify the pairwise relationship between two discrete event sequences. To achieve this, we resort to neural machine translation (NMT) models [23]. The intuition is that if we assign a distinct character to each discrete system state, then discrete event sequences are translated into a list of letters. We can then derive a sensor "language" by carefully portioning sequences of characters into words and sentences. With this transformation, NMT models are able to translate sentences of the source sensor to sentences of the target sensor, similar to translating from a source language to a target one. Translation quality is typically evaluated using the BiLingual Evaluation Understudy (BLEU) score [31] that can quantify the relationship between two discrete event sequences and eventually between the two sensors that produce these sequences. Based on this quantification, we are able to build the multivariate relationship graph, which can be used to discover system anomalies.

Our contributions are summarized as follows:

- We design a novel process to quantify pairwise relationships between discrete event sequences by leveraging NMT models originally designed for natural languages.
- We propose a methodology for discrete event sequences reported by sensors in real-world systems.
  - This methodology is *generic*. It directly works on categorical variables and does not rely on domain knowledge, thus it can be readily applied to any system with discrete event sequences.
  - This methodology is *unsupervised*. By capturing pairwise relationships among sensors during normal system operation, we build a multivariate relationship graph that provides useful information for knowledge discovery and anomaly detection.

We show the effectiveness of the proposed methodology on a real world system log of a physical plant that consists of 90% discrete event sequences. We further validate the proposed methodology with a public dataset of hard disk drive (HDD) failures [1] that illustrates how the methodology can work for continuous time series that are discretized.

II. Methodology

In this section we describe the data preparation process, the algorithms used to construct the pairwise relationship among the multivariate discrete event sequences, and how this information can be used for knowledge discovery and anomaly detection.

Figure 1 gives an overview. The left two boxes present the process of transforming multivariate discrete event sequences into multiple sensor "languages" composed of many "sentences". We then leverage NMT models to quantify the relationships of two sensors by translating sensor "languages" from one to the other. The next step is to build a multivariate relationship graph using the pairwise relationships among sensors. This relationship graph is built based on this quantification (i.e., translation scores) and provides useful information for knowledge discovery (i.e., exploring structures among various system components) and system-level anomaly detection (i.e., whether the system encounters an abnormal state).

A. Construction of Multivariate Relationship Graph

The multivariate discrete event sequences are composed of multiple event sequences collected from multiple sensors which are defined as $\{\mathbf{X}_t^k, k \in [1, 2, ..., \mathbf{N}], t \in [1, 2, ..., \mathbf{T}]\}$, where $\mathbf{N}$ is the total number of sensors, $\mathbf{T}$ is the total length of the training sequences, and $\mathbf{X}_t^k$ is record of a single event for sensor k (see the leftmost box in Figure 1). $\mathbf{X}_t^k$ captures for sensor k at time t the sensor's state: e.g., "on" and "off" for a switch or "status 1", "status 2", "status 3" for a recording unit, a command execution (i.e., "open" and "close" for a valve), or any other categorical information. The sensor output is evenly sampled which implies that the intervals between any two consecutive samples are the same. The cardinality of each event sequence is limited as the states of the sensor status are limited.

1) Sensor Encryption: Each event record is processed and converted into a standard coding schema in order to transform the sensor event sequences into different "languages" using the following steps:

- **Sequence Filtering:** The first step removes meaningless event sequences. If all events in a sequence are identical

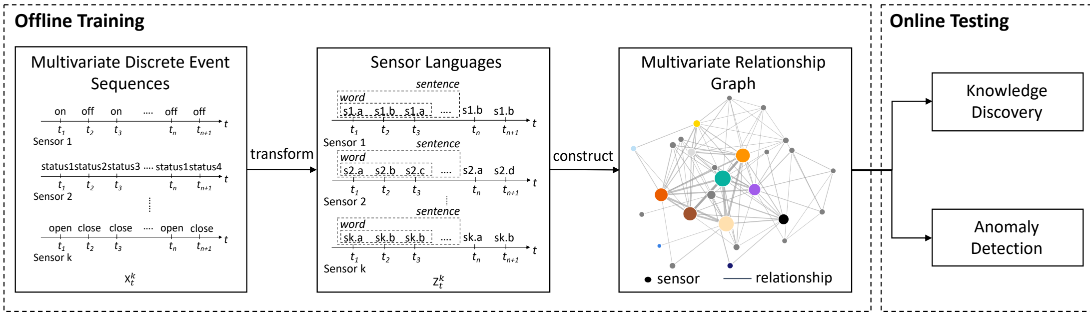


Fig. 1: Multivariate Discrete Event Sequence Analytics Framework.

during the entire sampling period, then this sequence cannot provide any contribution to the Language Translation Model (see section II-A3). We therefore exclude sequences with constant events, essentially we discard these sensors. Note that discarded sensors are not used in the online testing phase.

- **Discrete Event Encryption:** The second step encrypts each event record into a character. For each sequence, we collect the unique set of event records and sort them in alphanumeric order. Then we assign letters to each unique event record[1]. To differentiate across multiple sequences, we prefix the sensor name in front of the character. For example, the event record "on" in the sequence of sensor 1 is coded as "s1.a". The purpose of encryption is to map the event record into an alphabet so that the transformed sequence becomes a sensor language.

2) Language Sequence Generation: Once the encoded characters are obtained for each event record, we group the characters into words and sentences in order to leverage existing NMT models [23].

- **Converting Sequences to Words:** We compose "words" of equal length of i characters. Using a sliding window of j characters, we generate the next word. For example if $j = 1$, we use the first i characters of the sequence to compose the first word, the second to the $i + 1$th characters to compose the second word, and the third to the $i + 2$th characters to compose the third word, and so on. The distinct set of words derived by each sensor is the sensor vocabulary. One can choose meaningful values for i and j according to the dataset's sampling granularity. For a sample that uses per minute recording, $i = 10$ implies that each word contains the information of 10 minutes. If the sample uses per day recording, $i = 7$ implies that each word contains a week's worth of information. The value of j regulates the overlap between two consecutive words.
- **Word to Sentence Conversion:** We group words into sentences of equal length by setting the length of one sentence to m words with a sliding window of n words. The choice of m depends on the sampling granularity to include meaningful information. The choice of n decides the overlap of two consecutive sentences and determines the granularity of detection. For example, with a per minute sampling granularity and $n = 1$, detection can be performed every minute. Clearly, such fine-grained detection requires longer offline training time. The parameter n essentially controls the trade-off of the granularity of detection and training time and can be adjusted according to the prediction needs of the specific system.

[1]We always reserve a special character (i.e., <unk>) for any unknown system states which may occur in online testing.

Having transformed the multivariate discrete event sequences of all sensors ($\{\mathbf{X}_t^k\}$) into a corpus of sensor languages ($\{\mathbf{Z}_t^k\}$) (composed of words and sentences), we apply the NMT model to establish pairwise relationships among sensors.

3) Generation of the Language Translation Model: Algorithm 1 presents the main steps for generating the multivariate relationship graph $\mathbf{G}$. Given two language sequences from the multi-language corpus $\{\mathbf{Z}_t^k, k \in \mathbf{N}\}$, the algorithm applies the NMT model to build two *directional* pairwise relationships for each pair of language sequences (sensors). We therefore construct the multivariate relationship graph $\mathbf{G}$, where nodes represent sensors and edges represent the modeled relationships between sensors. Such graph models the inter-relationships among sensors in the system and can be used to deliver meaningful system-level knowledge.

Algorithm 1 Multivariate Relationship Graph Generation

Input: Multivariate Training Language Sequences $\{\mathbf{Z}_t^k,\ k \in [1, 2, ..., \mathbf{N}], t \in [1, 2, ..., \mathbf{T}]\}$
Output: Multivariate Relationship Graph $\mathbf{G}$
for $\{\mathbf{Z}_t^i,\ \mathbf{Z}_t^j\} \in \mathbf{Z}_t^k, i \neq j \in \mathbf{N}$ **do**
  $g(i, j) \leftarrow$ directional NMT model for (i, j) sensor pair
  $s(i, j) \leftarrow$ directional translation score for (i, j) sensor pair
  $\mathbf{G} \leftarrow g(i, j)$ and $s(i, j)$
return G

Neural Machine Translation (NMT) Model: The neural machine translation model uses a multi-layered Long Short-Term Memory (LSTM) to map the sentences of source language to a vector of a fixed dimensionality and uses another LSTM model to decode the vector into the sentences of target language [23], [37]. We transform the problem from originally extracting relationship of multivariate discrete event sequences to translating multi-lingual sensor "languages". We apply the NMT model to construct the nonlinear relationship $g(i,j)$ between each pair of sensors. Here, we use the state-of-the-art *seq2seq* model with attention mechanism [23].

Model Translation Score: The model translation score $s(i,j)$ quantifies the relationship between a pair of sensors (i,j). Here, we use BiLingual Evaluation Understudy (BLEU) score [31], the most commonly used metric to quantitatively evaluate the quality of machine translations. The score ranges from 0 to 100. A higher value indicates a better translation. By using the same architecture and parameter settings to train the NMT models for all sensor pairs, the BLEU scores are comparable across different pairs.

B. Knowledge Discovery

The multivariate relationship graph $\mathbf{G}$ produced by Algorithm 1 provides useful knowledge and insights for system-wide and component-wise relationships among different sensors: **(a)** discovery of system components (i.e., clusters of sensors) and **(b)** extraction of relationships between system components. Two types of subgraphs can be extracted from the multivariate relationship graph $\mathbf{G}$: **global subgraphs** and **local subgraphs**. These graphs can provide meaningful information on the system. For example, to discover structural information such as cluster structures of local sensors, we can apply to these graphs a random walk-based community detection algorithm [33]. Sensors in each local cluster likely originate from the same system component.

Clusters of sensors can be either isolated or connected. The sensors and edges connecting two clusters are critical as they are potentially responsible for error propagation after an anomaly occurs. Such component-wise knowledge offers root cause localization and insights on anomaly propagation for fault diagnosis. Section III-B provides a detailed discussion using a case study.

C. Anomaly Detection

Once the multivariate relationship graph $\mathbf{G}$ is generated through offline training, it can be used for online anomaly detection. Assuming that there is no anomaly in the training data, $\mathbf{G}$ represents system behavior under normal operation. Our focus is on detecting system anomalies when multiple sensors behave abnormally and lead to different relationships/interactions from those modeled in the offline training period. To this end, we define a metric called *anomaly score* to quantify the "significance" of anomalies (see definition in the next paragraph). Algorithm 2 presents the main steps for detecting an anomaly.

Given the multivariate testing language sequences $\{\mathbf{Y}_t^k, k \in \mathbf{N}\}$ and the graph $\mathbf{G}$, an anomaly is detected at time stamp t by applying the directional NMT model to the sensor pair (i,j) provided that the NMT model $g(i,j)$ is a valid one. The validity of NMT model $g(i,j)$ is determined by the range of BLEU score $s(i,j)$ set by the user [2]. A *broken relationship* between sensor pair (i,j) at time stamp t is detected if the testing BLEU score $f(i,j)$ is smaller than the BLEU score $s(i,j)$ obtained by training (see Algorithm 1).

Algorithm 2 Anomaly Detection

Input: Multivariate Testing Language Sequences
$\{\mathbf{Y}_t^k,\ k \in [1,2,...,\mathbf{N}], t \in [1,2,...,\mathbf{L}]\}$,
Multivariate Relationship Graph $\mathbf{G}$
Output: System Anomaly Score $\{\mathbf{a}_t,\ t \in [1,2,...,\mathbf{L}]\}$,
Sensor Pair Alert Status $\{\mathbf{W}_t,\ t \in [1,2,...,\mathbf{L}]\}$
for t in $[1,2,...,\mathbf{L}]$ **do**
  $\mathbf{a}_t \leftarrow 0,\ \mathbf{p}_t \leftarrow 0,\ \mathbf{W}_t \leftarrow 0,$
  for $\{\mathbf{Y}_t^i,\ \mathbf{Y}_t^j\} \in \mathbf{Y}_t^k, i \neq j \in \mathbf{N}$ **do**
    if directional NMT model $g(i,j)$ is a valid model **then**
      $\mathbf{p}_t \leftarrow \mathbf{p}_t + 1$
      $f(i,j) \leftarrow$ output BLEU scores by applying directional NMT model $g(i,j)$ for (i,j) sensor pair
    if $f(i,j) < s(i,j)$ **then**
      $\mathbf{a}_t \leftarrow \mathbf{a}_t + 1$
      $\mathbf{W}_t(i,j) \leftarrow 1$
  $\mathbf{a}_t \leftarrow \mathbf{a}_t / \mathbf{p}_t$
return $\mathbf{a}_t,\ \mathbf{W}_t$

One broken relationship may not be effective to detect anomalies in a complex real-world system. Therefore, we use the *anomaly score* to aggregate all broken relationships in the system. The anomaly score $\mathbf{a}_t$ is computed as the total number of broken relationships normalized by the total number of valid models (i.e., $\mathbf{p}_t$). Clearly, a larger score implies that more sensors behave abnormally. In addition, the sensor pair alert status $\{\mathbf{W}_t,\ t \in \mathbf{T}\}$ captures any link between two sensors with a broken relationship. This information ($\mathbf{a}_t$ and $\mathbf{W}_t$) can be used for interpreting anomalies.

III. Case Study I: Physical Plant Dataset

The dataset of the first case study is a proprietary one collected from a physical plant [3]. We demonstrate results for a publicly available dataset in Section IV.

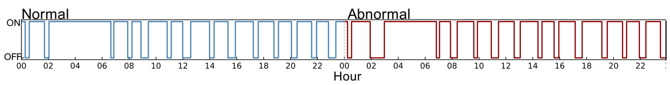
A. Dataset and Experiment Setup

The dataset is collected from a physical plant in November 2017 (30 days). During the entire month, there are only two days that are labeled post hoc as anomalous by the data owner: November 21 and 28. The limited number of anomalies in the dataset mandates the use of an unsupervised technique to learn the system behavior on normal days and use the learned knowledge to detect outliers as anomalies. There are 128 sensors recording system status in categorical format. Figure 2 shows the discrete event sequences recorded by two representative sensors in the system on one normal day and on

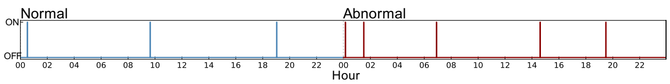
[2] Models with different BLEU scores show different detecting ability. In the evaluation sections (Section III-C and Section IV-D2), we find that models with BLEU scores in the [80, 90) range are best for anomaly detection.

[3] Unfortunately, we are not allowed to release the physical plant data log due to non-disclosure agreements.

one abnormal day. Both sensors report binary states, i.e., "ON" and "OFF". Sensor #4 (see Figure 2 (a)) exhibits periodical state changes while Sensor #91 (see Figure 2 (b)) mostly stays in the "OFF" state and occasionally switches to "ON". Despite the different behaviors of the two sensors, it is challenging to visually distinguish status changes between normal and abnormal days. *The purpose of the proposed methodology is to detect subtle state changes (if any) across pairs of sensors that are indicators of abnormal system operation.*



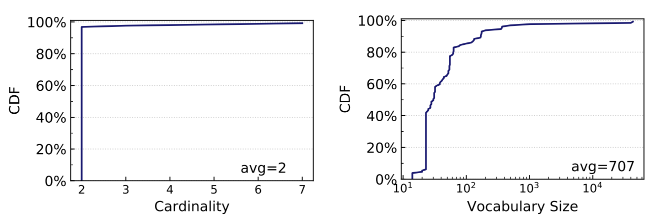
(a) Sensor #4



(b) Sensor #91

Fig. 2: Discrete event sequences collected by two representative sensors on one normal day (marked with blue) and one abnormal day (marked with red). System states are recorded every minute.

On average, sensors report 2.07 distinct discrete variables (i.e., system states). The majority (i.e., 97.6% of them) have a cardinality of 2, the one with the highest cardinality has 7 distinct discrete variables, see the CDF of sensor cardinalities in Figure 3 (a). With a log granularity of one minute, each sensor contains $30 \times 24 \times 60 = 43,200$ samples, resulting in a total of 5.5 million samples.



(a) CDF of sensor cardinality (b) CDF of vocabulary size

Fig. 3: (a) CDF of sensor event cardinality and (b) CDF of sensor vocabulary size.

1) Converting Discrete Event Sequences to Languages: Following the steps outlined in Section II-A2, we assign distinct characters (i.e., *a*, *b*, *c*, ...) to each categorical value, combine successive characters into words, and combine successive words into sentences. The sliding window size for words and sentences controls the overlap between neighboring words and sentences and regulates the total corpus size.

Generating words. We assume that 1 word consists of 10 characters. Consequently, each word contains the sensor status in the current minute (represented by the last character) and the history of the previous 9 minutes (as captured by the 9 preceding characters). We choose the size of the sliding window to be 1 character. With this, adjacent words have an overlap of 9 characters, adding one new character to represent the current system status.

We opt for long words (10 characters) because most sensors output two values only. Longer words result in a larger vocabulary size, passing more information to the translation model. Yet, the larger the vocabulary size, the longer the training time. A word size of ten characters strikes a good balance between prediction effectiveness and time efficiency. Figure 3(b) shows the CDF of the vocabulary size when the word size is 10. About 40% of sensors have vocabulary size smaller than 13, implying that the system states recorded by those sensors are stable for most of the time with only occasional changes (e.g., sensor #91 in Figure 3 (b)). Less than 20% of sensors have large vocabulary size (i.e., greater than 100). The average vocabulary size is 707 across all sensors.

Generating sentences. Each sentence contains 20 words that capture the system state changes for a period of 20 consecutive minutes. In contrast to words, there is no overlap between two consecutive sentences, i.e., the sentence sliding window is set to 20 words. This places the time granularity for anomaly detection at 20 minutes. The choice of the sentence length allows for a sufficiently tight time granularity of prediction without excessive training time.

Considering that every day each sensor records $24 \times 60 = 1440$ discrete values (or characters), there are $1440 \div 20 = 72$ sentences per sensor per day. In total, there are 276K sentences across all sensors. If more sentences are required (e.g., to refine the prediction granularity), the sliding window size can be decreased. For example, if the sentence sliding window size is set to be 1 word, then there are 1440 sentences per sensor per day, therefore anomaly detection is performed every minute. Note that such larger language corpus contains finer information on system state changes, but also results in longer model training time.

2) Model Settings: We train the model using events from normal, non-anomalous days. We use the first 10 days as the training set, the following 3 days as the development set, and the remaining 17 days for testing[4]. Note that both training and development sets consist of normal days only, the two anomalies are located in the test set.

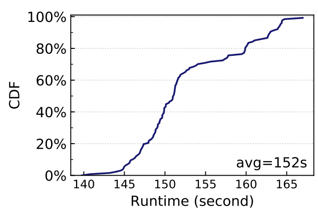
Recall that we leverage the NMT translation scores to quantify the strength of pairwise relationships between sensors. It is important to note that we have a strong preference toward NMT models that are able to *distinguish strong and weak pairwise relationships between sensors*, rather than models that are capable of delivering good translations even for bad cases (e.g., sentences with "grammar errors" or abnormal event sequences). We train two NMT models (directional) per sensor pair, essentially "translating" the language of the source sensor to that of the target sensor. Recall that, we use these models not for achieving high translation accuracy, but for quantifying the strength of the relationship between pair of sensors. The

[4]For splitting training/development/testing datasets, we tried various partitions with results of similar quality. We therefore opted for a small training set to allow for more testing cases.

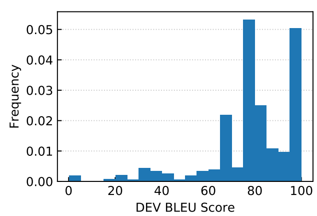
two derived translation scores for each pair reflect the strength of their relationship. Higher scores imply that the two sensors are more related.

We apply the same parameter settings to each NMT model to ensure that all pairwise relationships are quantified with the same metric. Here, we use the state-of-the-art NMT model [23] and set the parameters of each model as follows: # of LSTM layers=2, # hidden units=64, # embedding size=64, # training steps=1000, and dropout=0.2. We tried different settings but the above one is deemed suitable as it delivers good distinguishing ability while maintaining acceptable training time.

Figure 4 (a) shows the CDF of model runtime. On average, each model requires 2.5 minutes for both training and testing. Therefore, model scalability is not a concern. This can be further accelerated if this process is done in parallel for different sensor pairs. Moreover, by comparing the pattern of sensor discrete event sequences, we notice that many sensors actually share similar event sequences. If redundant sensors are further filtered out, then models are trained on representative sensors only and training time reduces significantly. Since our aim here is to capture the state of the entire system, we run all pairwise models.



(a) CDF of model runtime



(b) Histogram of BLEU scores

Fig. 4: (a) CDF of model runtime including training and testing time and (b) Histogram of the BLEU scores.

B. Multivariate Relationship Graph

After completing the training process, we use the development set (3 normal days) to collect translation (BLEU) scores for all sensor pairs. These scores serve as measures that quantify the strength of the relationship between the discrete event sequences of two sensors during normal operating conditions. Higher scores imply that the two sequences are similar; lower scores imply the opposite. Figure 4 (b) shows the histogram of the BLEU scores. Notice that the majority (i.e., 89.4%) of scores are greater than 60, implying that the discrete event sequences of most sensors are related. Intuitively, one may surmise that stronger relationships are preferable for knowledge discovery and anomaly detection, since it is more readily apparent when these relationships are violated. After trying different score ranges, we find that relationships with scores in the [80, 90) range provide the most accurate information[5]. We provide evidence for this in the rest of this section.

1) Global Subgraphs : If we treat each sensor as a node and the relationship between each pair of sensors as an edge, we can obtain a *directed* graph of all sensors representing the system (i.e., the multivariate relationship graph **G** returned by Algorithm 1). There are two edges between any two sensors, each edge representing a "translation" from the source language to the target language. The BLEU score acts as an edge weight representing the quality of the "translation". Note that the same BLEU score of the edges that connect the same two sensors may be different. This full graph is the *original* multivariate relationship graph (short as Ori-MVRG). If we were to plot all sensors and their relationships in the Ori-MVRG, the graph would be fully connected, though the relationships would be of varying strengths. Such a graph is too noisy to be useful. We therefore partition the Ori-MVRG into subgraphs, according to edge weights (i.e., BLEU scores of sensor pairs). We choose a set of score ranges and produce a subgraph for each range. A given edge is included in a subgraph if and only if its BLEU score falls into the corresponding range. If a sensor has no edges in a subgraph, it is deleted from that subgraph. Table I shows the partition results according to the selected ranges of BLEU score ranges. We merge less significant subgraphs (i.e., relationships with scores smaller than 60) into a single subgraph .

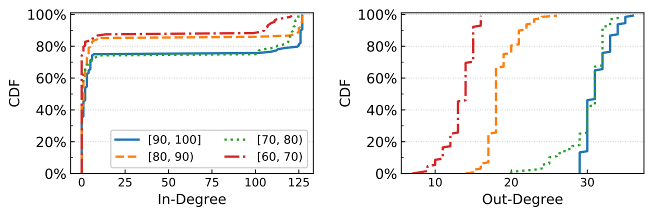
TABLE I: Statistics for global subgraphs at different BLEU score ranges.

BLEU score range	[0, 60)	[60, 70)	[70, 80)	[80, 90)	[90, 100]
% relationships	10.6%	12.8%	28.8%	17.8%	29.9%
# sensors	54	32	56	73	82
# popular sensors (in-degree $\geq$ 100)	9	14	32	18	31
# relationships (w/o popular sensors)	344	162	77	146	151

Figure 5 shows the CDFs of in-degree and out-degree for the subgraphs defined in Table I. Looking at the in-degree (see Figure 5 (a)), we notice that a small portion (around 20% to 25%) of sensors are "popular", i.e., with in-degree $\geq$ 100, while others are much less connected to, i.e., with in-degee $\approx$ 10. Table I lists the number of popular sensors in each global subgraph in its third row. These sensors are critical indicators of system health status as any abnormal behaviors in their event sequences would propagate broadly. Figure 5 (b) shows that the out-degree distribution spreads relatively evenly, falling between 10 to 35. To visually show a global subgraph , we plot the one defined by the [80, 90) score range in Figure 6. The larger nodes in the figure signify the popular ones (i.e., with in-degree ≥ 100).

2) Local Subgraphs : The example of the global subgraph illustrated in Figure 6 is still too densely connected to provide useful clustering information across sensors. We therefore remove the popular sensors (i.e., those with in-degree ≥ 100) from each global subgraph to generate the corresponding local subgraphs. Figures 7 (a) and 7 (b) show the local subgraphs for

[5]This optimal range also applies to other datasets, including the Backblaze dataset, see Section IV-D.



(a) CDF of in-degree (b) CDF of out-degree

Fig. 5: CDFs of (a) in-degree and (b) out-degree of sensors in global subgraphs at different BLEU score ranges.

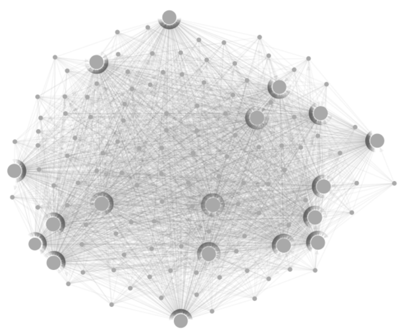
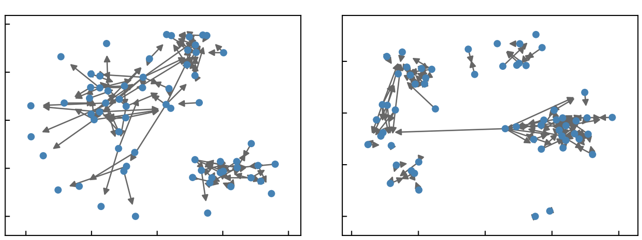


Fig. 6: Global subgraph with BLEU score in the [80, 90) range. Each node represents a sensor and each edge represents the relationship between two sensors. Larger nodes correspond to popular sensors with in-degree ≥ 100.

[80, 90) and [90, 100] score ranges, respectively. Both figures clearly illustrate the presence of several clusters of sensors. In addition, most clusters are isolated, i.e., not connected to others, although there is an exception where we see some loose connectivity, see Figure 7 (a) where two clusters are connected through a single edge.



(a) Local subgraph at [80, 90) (b) Local subgraph at [90, 100]

Fig. 7: Local subgraphs with BLEU score in the (a) [80, 90) and (b) [90, 100] range.

In all cases, the clusters reflect the underlying system architecture. Sensors in the same cluster could come from same system components or record similar or related system states. This assumption is confirmed by domain experts on this dataset. Furthermore, this clustering information is extremely useful as some data are often encrypted and provide no information about the organization of the real system. Yet, the local subgraphs allow us to surpass this difficulty and identify related sensors.

Knowledge Discovery Takeaways:

- Global subgraphs identify sensors that are critical indicators of system health status.
- Local subgraphs identify sensors that come from the same system components or report similar system states.

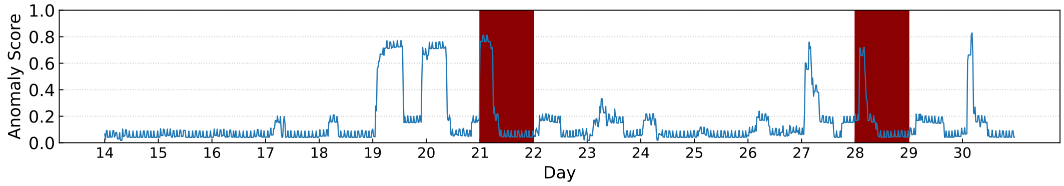
C. Anomaly Detection

In this section, we discuss how anomalies are detected in the online testing component. Here, we calculate the anomaly score (see Section II-C for its definition) for every sentence in the testing dataset. Higher anomaly scores indicate that more pairwise relationships are broken, thus an anomaly is detected with higher confidence.

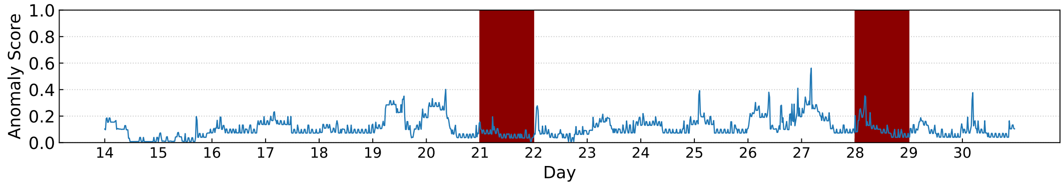
Global subgraphs are superior to local ones for anomaly detection, the reason is that popular sensors contain more information about sensor interactions during normal times and are critical indicators of system health status. Therefore, we focus on global subgraphs in the rest of this section.

Figure 8 shows the anomaly detection results with global subgraphs at two different BLEU score ranges. The x-axis is the timeline and the y-axis shows the calculated anomaly score. Recall that the anomaly score reflects how many pair-wise sensor relationships are broken at a certain timestamp. There are two anomalies in the testing dataset (marked with red shade in Figure 8). The global subgraph with BLEU score in the [80, 90) range delivers the most clear detection (see Figure 8 (a)), as it correctly detects the anomalies of days 21 and 28 (i.e., the anomaly score is close to 0.8). The other four spikes are false positives. However, we observe that they closely precede the two anomalies. As confirmed with the domain experts, the spikes before the two true anomalies (days 19, 20, and 27) are sings of early detection, which could signal system administrators to take proactive actions. During normal times, the anomaly scores are low, mostly below 0.2. Notice that, on the 30th day, there is also a sign of anomaly (i.e., score over 0.8). If the following day were to be confirmed to be an anomaly (unfortunately we do not have log data beyond day 30), then it would be a correct sign of early detection. Otherwise, it is a false positive.

Figure 8 (b) illustrates the anomaly detection results for the global subgraph with BLEU score in the [90, 100] range, i.e., the subgraph with the strongest relationships. In this case, the anomaly scores are too low to give clear signs of anomalies. To better understand why the global subgraph with the strongest relationships fails, we look closely at the translation results of the target sensors and notice that these sensors have simple languages. Their system states barely change over the entire month, resulting in a small vocabulary size. For example, a significant portion of words in the vocabulary of these target sensors are "*aaaaaaaa*" (due to unchanged state over an extended period). Translating to sentences composed only of that simple word would inevitably result in high translation scores. In other words, the global subgraph with BLEU score in [90, 100] range does not necessarily contain sensors that



(a) Global subgraph with BLEU score in [80, 90) range



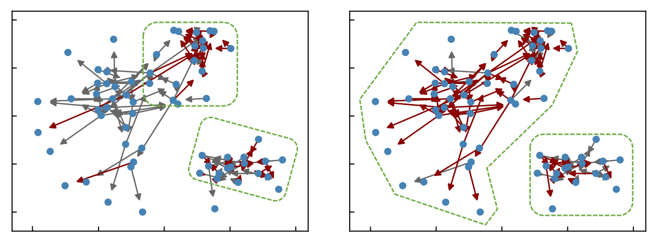
(b) Global subgraph with BLEU score in [90, 100] range

Fig. 8: Anomaly detection with global subgraphs with different BLEU score ranges.

are strongly related but instead clusters of easily translatable sensors.

We experimented with the remaining of global subgraphs, results are not presented here due to lack of space and can be summarized as follows: Global subgraphs of weaker relationships (i.e., with BLEU score lower than 80) generally do well but can result in many false positives.

Interpretation of anomaly detection results: For each detected anomaly, the framework uses the local subgraphs for fault diagnosis by tracing the broken relationships in the multivariate relationship graph and identifies problematic sensors that are responsible for the anomaly. Figure 9 presents the fault diagnosis results for the anomaly detection illustrated in Figure 8 (a). Red edges in Figure 9 represent broken relationships, i.e., the BLEU score is lower than that that of normal times (see Section II-C for more details). Gray edges correspond to normal relationships, as these sensor pairs behave normally even during anomalous days. The broken relationships can be used to locate sensors that should be responsible for the corresponding anomaly. In Figure 9 (a), we observe that sensors clustered in the upper right and lower right corners (i.e., circled with green lines) are problematic. In contrast, in Figure 9 (b), almost all relationships are broken (i.e., see the red edges circled in green boxes), implying that this is a severe anomaly that affects a significant portion of sensors. Such diagnosis is helpful to system administrators to quickly locate problematic sensors and the source of the anomaly. The analytics framework provides the option to describe similar figures for each anomaly at finer granularities, e.g., every hour, to visually present how faults propagate through sensors over time. Results are not shown here due to space constraints.



(a) 2017-11-21 (b) 2017-11-28

Fig. 9: Fault diagnosis with local subgraphs with BLEU score in [80, 90) range on anomalous days. Edges marked with red are broken relationships. Green circles indicate faulty clusters of sensors that are responsible for the anomalies.

Anomaly Detection Takeaways:

- Global subgraphs are more suitable for anomaly detection than local subgraphs .
- Global subgraphs with the strongest relationships (i.e., BLEU score above 90) are not useful.
- Local subgraphs are helpful to locate sensors that are responsible for anomalies.
- Fault diagnosis can be performed at various time granularities to show fault propagation over time.

IV. Case Study II: HDD Dataset

To provide verifiable results, we apply here the proposed analytics framework to a publicly available dataset. Unfortunately, among the public datasets with categorical data in the Kaggle repository [2], none reports failures or anomalies. Thus, we chose to use the Backblaze dataset that is publicly available and details HDD reliability statistics (disk failures) over time [1]. The Backblaze data set has mostly continuous features but with some minimal preprocessing we show that they may be converted into discrete ones.

A. Dataset of HDD Failures

The Backblaze data consist of daily performance logs collected from hard disk drives (HDDs) housed at the Backblaze data center. For each day of operation, the drives report a list of SMART attributes [3], which report either cumulative lifetime counts of particular hardware events (e.g., certain types of drive errors) or daily summaries of drive activity (e.g., read or write counts). Of particular interest to us is an additional attribute marked by the maintainers of the Backblaze dataset: certain days of drive operation are marked as "failures," indicating that, on the following day, the drive ceases to function and is removed from production.

B. Baseline Models

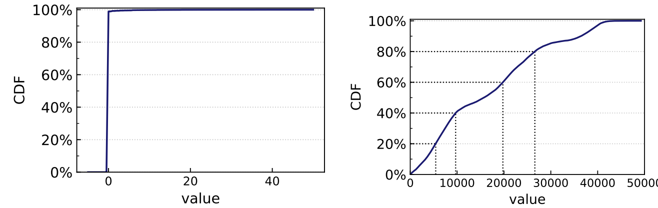
We start with choosing the best set of features. Backblaze claims to report 100+ SMART features, but many of them are only recorded on a small proportion of drives or drive models. Therefore, we restrict ourselves to features that are recorded for all disk drive types, resulting in 20 features only. In addition, we find that 14 of these features are cumulative counts, monotonically increasing over the lifetime of the drive. These features change relatively slowly over time and due to their trending behavior, they tend to spuriously correlate with other cumulative features. Due to the difficulties this poses to our correlation-based approach, we transform these cumulative features using a first-order difference, yielding a set of daily deltas. In the end, we utilize 34 features, including 20 raw SMART features and 14 differenced ones. We consider the latest data collected in 2018 and focus on data reported by Seagate models for enterprise workloads, which account for 35% of the Backblaze data in total and 46% of the data for Seagate models in 2018.

To obtain a performance baseline, we use two commonly used nonlinear machine learning models:

- **Random Forest (RF).** This is a supervised ensemble model based on decision trees, which has been previously successfully applied to the Backblaze dataset [24]. We use 80% of the drives for training and 20% for testing. Since there are many more non-failures than failures, we randomly sub-sample non-failure cases such that the training data has a 1-to-1 majority-to-minority ratio.
- **One-class SVM (OC-SVM).** This is a widely used unsupervised model for anomaly detection [35]. Here, we choose the radial basis function (RBF) kernel. The OC-SVM takes as training data a set of non-anomalous observations and fits a decision boundary about these data. If a drive in the training set is not observed to fail, we consider its data "non-anomalous". We find that training the OC-SVM scales poorly to large datasets, so we randomly sub-sample from among these relevant observations to get a more manageable training set.

C. Multivariate Relationship Graph

In order to adapt our analytics framework to the Backblaze dataset, each SMART feature is taken to be recorded by a sensor. In the multivariate relationship graph, each node represents a feature fed into the framework. As a generic method that does not require feature engineering, our framework accepts raw SMART features only. In addition, among the 20 raw features, the values of 4 features are barely changed in the year. As discovered in Section III-B, these features provide little information, thus are removed. So, there should be 16 nodes (i.e., features) and $16 \times 15 = 240$ edges (i.e., relationships) in the learned multivariate relationship graph.



(a) SMART 187 (b) SMART 9

Fig. 10: Examples of features used for the two feature discretization schemes, shown as CDFs.

Unlike the physical plant system where sensors continue to collect information when an anomaly happens, disks in Backblaze are removed at the time they fail. As a result, each disk only reports one failure sample, which is the last day of its operation. In order to acquire more anomalies (rather than only 1), we aggregate the data for all disks so that the number of anomalies corresponds to the number of failure disks.

Recall that our method is designed for discrete event sequences. To make the Backblaze dataset amenable to our model, we have to discretize the continuous values recorded by those features. We tailor our discretization schemes in this way in order to best maintain the semantics of each feature. Figure 10 illustrates the two discretization schemes used with two representative examples.

1) If most of the observations of a feature are equal to zero (as is the case with many error counts), then we adopt a binary discretization scheme. We replace the feature with an indicator variable, indicating whether the feature is zero (see Figure 10 (a)).
2) For all other features, we examine the distribution of the feature across the training set and pull out the 20th, 40th, 60th, and 80th percentiles. These are used as decision boundaries for assigning categories (see Figure 10 (b)).

To ensure a stable discretization of features, we focus on disks with substantial samples. Here we consider disks with over 10-month data in the year, ending up with 24 disks. For each disk, we utilize its last 4 months data: the first 2 months for training, the following one month for development, and the last one for testing. For NMT model parameter settings, we use the ones used for the private dataset (see Section III-A2). Since features are recorded on a daily-basis, we set 1 word to 5 characters and 1 sentence to 7 words (both sliding windows are set to be 1) to ensure a reasonable vocabulary size and number of sentences.

D. Evaluation

Our analytics framework is a generic unsupervised method that is especially designed for discrete event sequences. Leveraging the learned multivariate relationship graph, it provides information about important features (i.e., reflected as nodes with higher in-degree in global subgraphs, see Section IV-D1) and disk failures (i.e., anomaly detection, see Section IV-D2).

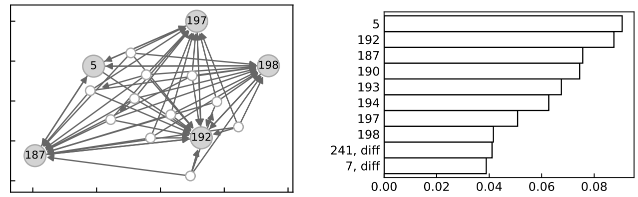
The graphs can also help in locating features for fault diagnosis but not shown here due to space constraints. In contrast, the baseline models, which do not work directly for discrete event sequences and rely on domain knowledge for feature engineering, ether require more information as a supervised method (i.e., Random Forest) or cannot provide feature importance analysis (i.e., one-class SVM). Table II shows a high-level comparison among the models. Note that our method is *unsupervised, generic* and especially designed for discrete event sequences where RF and OC-SVM are not applicable. It is therefore expected for our method to not achieve as high recall comparing to algorithms for continuous time series that rely on feature engineering.

TABLE II: Comparison of different models on Backblaze.

Model	Unsupervised?	Feature Engineering?	Feature Ranking?	Recall	Applicable to Discrete Event Sequences?
RF	×	✓	✓	70%-80%	×
OC-SVM	✓	✓	×	60%	×
Ours	✓	×	✓	58%	✓

1) Knowledge discovery: Let us start with knowledge discovery with global subgraphs. Figure 11 (a) shows the global subgraphs with BLEU score in [80, 90) range. This range works best for the power plant dataset (see Section III-B) and continues to be the best for Backblaze also. In the graph, there are 16 nodes representing 16 features. 5 of them are labeled with their SMART feature ID (i.e., larger nodes in the figure). These features are more extensively connected to others, implying that they are critical indicators of disk health status. Table III lists the descriptions of those features and their in-degree and out-degree. Nonzero values for these features indicate that the disk has suffered failed I/O operations and that the health of the disk is at risk. Their importance to anomaly detection aligns with what we expect to be necessary for prediction. As comparison, Figure 11 (b) presents 10 most important features given by the feature importance score provided by the Random Forest model. We observe that all 5 features are shown in the list and we find that these features consistently place in the top 10 upon model retraining. This confirms the effectiveness of using global subgraphs for feature importance analysis.

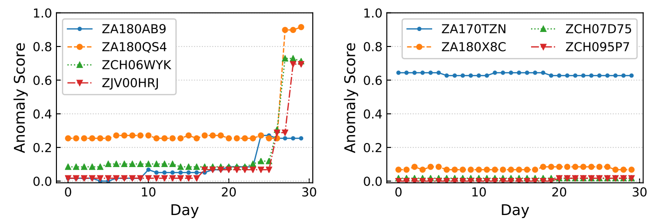
2) Anomaly detection: We use the global subgraphs shown in Figure 11 (a) to detect disk failures in Backblaze. Here, we select several failed disks that are successfully detected and several that are not detected, and show how their anomaly scores change over time before their failure dates, see Figure 12. From Figure 12 (a), we observe that there is always a sharp increase in the anomaly score (i.e., over 0.5 increment) right before the failure date for every successfully detected disk. In contrast, in Figure 12 (b), the anomaly scores for non-successfully-detected ones are stable over time, regardless at high scores of over 0.6 or at low scores of below 0.1. In other words, we look for sharp increments as signs of disk



(a) Global subgraph (b) Feature ranking by RF

Fig. 11: Feature importance analysis (a) by global subgraph with BLEU score in the [80, 90) range and (b) by the Random Forest model.

failures. In total, we are able to deliver a recall of 58%, which is comparable to the recall of 60% given by the one-class SVM trained on more features, without any feature engineering effort that one-class SVM requires.



(a) Detected disks (b) Not detected disks

Fig. 12: Disk failure detection given by global subgraph with BLEU score in the [80, 90) range.

V. Related Work

Understanding the interdependence relationship among multivariate time series is one of the most important tasks in machine learning [14]. Most of existing literature focuses on time series analysis of continuous sequences [40]–[42]. The canonical correlation analysis [15] and its nonlinear version of kernel canonical correlation analysis [5] aim at extraction of common features from a pair of multivariate sequences based on a linear or nonlinear transformation of the original variables by maximization of correlation or kernelized correlation [5], [15]. There are multiple dependence measures to quantify bivariate relationships such as Spearman's ρ, Spearman's footrule, Gini's coefficient, Kendall's τ [19], copula-based Kernel dependency measures [32], kernel dependency estimation [38] and others. In time series analysis, auto-regressive and moving average (ARMA) models have been proposed to characterize the dependence measure between two time series [14]. ARMA builds a linear relationship model between two continuous time series. For example, a method based on an ARMA model was proposed to quantify multivariate invariant relationship in a distributing system [36]. These methodologies work well for continuous time series and variables, but have limitations when applied to discrete event sequences.

Regarding discrete categorical event data analysis, some works focus on mining patterns from event sequences. The discovered patterns can be used for further analysis, such as

TABLE III: Top 5 most important SMART features [3] reported by global subnetworks at [80, 90).

ID	Name	# in-degree	# out-degree	Description
192	Power-off Retract Count	15	3	Number of power-off or emergency retract cycles.
187	Reported Uncorrectable Errors	13	2	The count of errors that could not be recovered using hardware ECC.
198	(Offline) Uncorrectable Sector Count	13	2	The total count of uncorrectable errors when reading/writing a sector. A rise in the value of this attribute indicates defects of the disk surface and/or problems in the mechanical subsystem.
197	Current Pending Sector Count	13	2	Count of "unstable" sectors (waiting to be remapped, because of unrecoverable read errors).
5	Reallocated Sectors Count	3	4	Count of reallocated sectors. The raw value represents a count of the bad sectors that have been found and remapped. This value is primarily used as a metric of the life expectancy of the drive.

anomaly detection. Algorithms that detect frequent episodes in event sequences can be found in [20], [25]. Hubballi *et. al.* [16] leverage the concept of n-gram in natural language processing to model patterns in system calls, by constructing n-gram trees for short sequences of system calls. The above works share similarities with our method in terms of utilizing a sliding window to cut the discrete events into many sub-sequences. However, these works apply on event sequences of high cardinality while here we are restricted by the fact that all sequences are of low cardinality.

There are also works that focus on anomaly detection in a *single* discrete categorical event sequence [26]. Dong *et. al.* propose an efficient method for mining emerging patterns in a categorical data sequence for trend discovery [12]. A method based on MaxCut is proposed to detect anomalous events in activity networks [34]. Another anomaly detection technique that focuses on categorical datasets [11] uses a local anomaly detector to identify individual records with anomalous attribute values and then detects patterns where the number of anomalous records is higher than expected. The above techniques are applied to a single discrete event sequence without considering the time aspect of events or without considering any inter-dependence relationship among different sources of events or variables. Therefore, these algorithms cannot discover component-wise knowledge on the underlying system or offer anomaly detection based on system-wide understanding and modeling. Multidimensional Hawkes processes [22] and its variants [13], [27] have been applied to model the inter-dependent relationship across multi-source events. Hawkes processes are based on point processes that model the conditions that affect the intensity of arrival events with changing intervals.

The proposed methodology in this paper is a generic unsupervised learning framework for discrete event sequences collected from sensors. By leveraging language translation, we successfully quantify pairwise relationships among sensors, which in turn are used to build a multivariate relationship graph. This graph is able to provide valued system information for knowledge discovery and anomaly detection.

VI. Conclusion

Modern physical systems deploy large numbers of sensors to record the system status of different components. Although a significant portion of those sensors report discrete event sequences, there is a lack of effective methodologies that are designed especially for such data. In this paper, we propose an unsupervised learning analytics framework specially designed for discrete event sequences collected from sensors in real-world systems for knowledge discovery, anomaly detection, and fault diagnosis. It leverages the concept of language translation by considering the discrete event sequences of sensors as their languages and then apply NMT models to translate the language of one sensor to another. Naturally, the translation score turns to be an effective metric to quantify the pairwise relationships among discrete event sequences of sensors, which are difficult to learn by most state-of-the-art algorithms that mainly work for continuous time series. With sensor pairwise relationships, the analytics framework builds a multivariate relationship graph to represent system information. The proposed framework is validated on two real-world datasets: a proprietary dataset from a physical plant and a public hard disk drive dataset from Backblaze, and is shown to be effective in extracting sensor pairwise relationships for knowledge discovery, anomaly detection, and fault diagnosis.

Acknowledgements

We thank our shepherd Patrick Lee. The majority of the presented work was completed during a summer internship of Bin Nie at NEC Labs. Smirni, Nie, and Alter are partially supported by NSF grants CCF-1649087 and IIS-1838022.

References

[1] Backblaze hard drive data. https://www.backblaze.com/b2/hard-drive-test-data.html.

[2] Kaggle datasets. https://www.kaggle.com/datasets.

[3] S.m.a.r.t. https://en.wikipedia.org/wiki/S.M.A.R.T.#ATA_S.M.A.R.T._attributes.

[4] S. Agrawal and J. Agrawal. Survey on anomaly detection using data mining techniques. *Procedia Computer Science*, 60:708–713, 2015.

[5] S. Akaho. A kernel method for canonical correlation analysis. In *Proceedings of International Meeting on Psychometric Society*, 10 2001.

[6] J. Alter, J. Xue, A. Dimnaku, and E. Smirni. SSD failures in the field: symptoms, causes, and prediction models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2019, Denver, Colorado, USA, November 17-19, 2019*, pages 75:1–75:14. ACM, 2019.

[7] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.

[8] M. M. Botezatu, I. Giurgiu, J. Bogojeska, and D. Wiesmann. Predicting disk replacement towards reliable data centers. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 39–48, New York, NY, USA, 2016. ACM.

[9] L. Cherkasova, K. M. Ozonat, N. Mi, J. Symons, and E. Smirni. Anomaly? application change? or workload change? towards automated detection of application performance anomaly and change. In *The 38th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2008, June 24-27, 2008, Anchorage, Alaska, USA, Proceedings*, pages 452–461. IEEE Computer Society, 2008.

[10] R. Chitrakar and H. Chuanhe. Anomaly detection using support vector machine classification with k-medoids clustering. In *2012 Third Asian Himalayas International Conference on Internet*, pages 1–5. IEEE, 2012.

[11] K. Das, J. Schneider, and D. B. Neill. Anomaly pattern detection in categorical datasets. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 169–176, New York, NY, USA, 2008. ACM.

[12] G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, pages 43–52, New York, NY, USA, 1999. ACM.

[13] N. Du, H. Dai, R. Trivedi, U. Upadhyay, M. Gomez-Rodriguez, and L. Song. Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1555–1564, New York, NY, USA, 2016. ACM.

[14] J. Hamilton. *Time series analysis*. Princeton Univ. Press, Princeton, NJ, 1994.

[15] H. Hotelling. Relations between two sets of variates. *Biometrika*, 28:321–377, 1936.

[16] N. Hubballi, S. Biswas, and S. Nandi. Sequencegram: n-gram modeling of system calls for program based anomaly detection. In *2011 Third International Conference on Communication Systems and Networks (COMSNETS 2011)*, pages 1–10. IEEE, 2011.

[17] G. Jiang, H. Chen, and K. Yoshihira. Modeling and tracking of transaction flow dynamics for fault detection in complex systems. *IEEE Transactions on Dependable and Secure Computing*, pages 312–326, 2006.

[18] X. Jin, Y. Guo, S. Sarkar, A. Ray, and R. M. Edwards. Anomaly detection in nuclear power plants via symbolic dynamic filtering. *IEEE Transactions on Nuclear Science*, 58(1):277–288, Feb 2011.

[19] H. Joe. *Multivariate Models and Dependence Concepts*. Chapman & Hall, 1997.

[20] M. Leemans and W. M. van der Aalst. Discovery of frequent episodes in event logs. In *International Symposium on Data-Driven Process Discovery and Analysis*, pages 1–31. Springer, 2014.

[21] T. Li, Y. Jiang, C. Zeng, B. Xia, Z. Liu, W. Zhou, X. Zhu, W. Wang, L. Zhang, J. Wu, L. Xue, and D. Bao. Flap: An end-to-end event log analysis platform for system management. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 1547–1556, New York, NY, USA, 2017. ACM.

[22] D. Luo, H. Xu, Y. Zhen, X. Ning, H. Zha, X. Yang, and W. Zhang. Multi-task multi-dimensional hawkes processes for modeling event sequences. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 3685–3691. AAAI Press, 2015.

[23] M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[24] F. Mahdisoltani, I. Stefanovici, and B. Schroeder. Proactive error prediction to improve storage system reliability. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)*, pages 391–402, Santa Clara, CA, 2017. USENIX Association.

[25] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data mining and knowledge discovery*, 1(3):259–289, 1997.

[26] E. McFowland, S. Speakman, and D. B. Neill. Fast generalized subset scan for anomalous pattern detection. *Journal of Machine Learning Research*, 14:1533–1561, 2013.

[27] H. Mei and J. Eisner. The neural Hawkes process: A neurally self-modulating multivariate point process. In *Advances in Neural Information Processing Systems*, Long Beach, Dec. 2017.

[28] B. Nie, D. Tiwari, S. Gupta, E. Smirni, and J. H. Rogers. A large-scale study of soft-errors on gpus in the field. In *2016 IEEE International Symposium on High Performance Computer Architecture, HPCA 2016, Barcelona, Spain, March 12-16, 2016*, pages 519–530. IEEE Computer Society, 2016.

[29] B. Nie, J. Xue, S. Gupta, C. Engelmann, E. Smirni, and D. Tiwari. Characterizing temperature, power, and soft-error behaviors in data center systems: Insights, challenges, and opportunities. In *25th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS 2017, Banff, AB, Canada, September 20-22, 2017*, pages 22–31. IEEE Computer Society, 2017.

[30] B. Nie, J. Xue, S. Gupta, T. Patel, C. Engelmann, E. Smirni, and D. Tiwari. Machine learning models for GPU error prediction in a large scale HPC system. In *48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2018, Luxembourg City, Luxembourg, June 25-28, 2018*, pages 95–106. IEEE Computer Society, 2018.

[31] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[32] B. Póczos, Z. Ghahramani, and J. G. Schneider. Copula-based kernel dependency measures. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress, 2012.

[33] P. Pons and M. Latapy. Computing communities in large networks using random walks. *Journal of Graph Algorithms and Applications*, pages 191–218, 2006.

[34] P. Rozenshtein, A. Anagnostopoulos, A. Gionis, and N. Tatti. Event detection in activity networks. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1176–1185, New York, NY, USA, 2014. ACM.

[35] B. Schölkopf and A. J. Smola. *Learning with Kernels: support vector machines, regularization, optimization, and beyond*. Adaptive computation and machine learning series. MIT Press, 2002.

[36] A. B. Sharma, H. Chen, M. Ding, K. Yoshihira, and G. Jiang. Fault detection and localization in distributed systems using invariant relationships. In *Dependable Systems and Networks (DSN), 2013 43rd Annual IEEE/IFIP International Conference on*, pages 1–8. IEEE, 2013.

[37] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112, Cambridge, MA, USA, 2014. MIT Press.

[38] J. Weston, O. Chapelle, V. Vapnik, A. Elisseeff, and B. Schölkopf. Kernel dependency estimation. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 897–904. MIT Press, 2003.

[39] J. Xue, R. Birke, L. Y. Chen, and E. Smirni. Managing data center tickets: Prediction and active sizing. In *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 335–346. IEEE, 2016.

[40] J. Xue, R. Birke, L. Y. Chen, and E. Smirni. Spatial-temporal prediction models for active ticket managing in data centers. *IEEE Trans. Network and Service Management*, 15(1):39–52, 2018.

[41] J. Xue, B. Nie, and E. Smirni. Fill-in the gaps: Spatial-temporal models for missing data. In *13th International Conference on Network and Service Management, CNSM 2017, Tokyo, Japan, November 26-30, 2017*, pages 1–9. IEEE Computer Society, 2017.

[42] J. Xue, F. Yan, R. Birke, L. Y. Chen, T. Scherer, and E. Smirni. PRACTISE: robust prediction of data center time series. In *11th International Conference on Network and Service Management, CNSM 2015, Barcelona, Spain, November 9-13, 2015*, pages 126–134. IEEE Computer Society, 2015.

[43] Y. Yasami and S. P. Mozaffari. A novel unsupervised classification approach for network anomaly detection by k-means clustering and id3 decision tree learning methods. *The Journal of Supercomputing*, 53(1):231–245, 2010.

[44] S.-J. Yen and Y.-S. Lee. Under-sampling approaches for improving prediction of the minority class in an imbalanced dataset. In *Intelligent Control and Automation*, pages 731–740. Springer, 2006.

[45] K. Zhang, J. Xu, M. R. Min, G. Jiang, K. Pelechrinis, and H. Zhang. Automated it system failure prediction: A deep learning approach. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 1291–1300. IEEE, 2016.

[46] S. Zhang, Y. Liu, W. Meng, Z. Luo, J. Bu, S. Yang, P. Liang, D. Pei, J. Xu, Y. Zhang, Y. Chen, H. Dong, X. Qu, and L. Song. Prefix: Switch failure prediction in datacenter networks. *POMACS*, 2(1):2:1–2:29, 2018.