2018 17th IEEE International Conference on Machine Learning and Applications

A Minimum Spanning Tree Clustering Approach for Outlier Detection in Event Sequences

Shahrooz Abghari, Veselka Boeva, Niklas Lavesson, Håkan Grahn
Department of Computer Science and Engineering
Blekinge Institute of Technology
Karlskrona, Sweden
Email: shahrooz.abghari@bth.se

Selim Ickin, Jörgen Gustafsson
Ericsson Research
Ericsson AB
Stockholm, Sweden

***Abstract*—Outlier detection has been studied in many domains. Outliers arise due to different reasons such as mechanical issues, fraudulent behavior, and human error. In this paper, we propose an unsupervised approach for outlier detection in a sequence dataset. The proposed approach combines sequential pattern mining, cluster analysis, and a minimum spanning tree algorithm in order to identify clusters of outliers. Initially, the sequential pattern mining is used to extract frequent sequential patterns. Next, the extracted patterns are clustered into groups of similar patterns. Finally, the minimum spanning tree algorithm is used to find groups of outliers. The proposed approach has been evaluated on two different real datasets, i.e., smart meter data and video session data. The obtained results have shown that our approach can be applied to narrow down the space of events to a set of potential outliers and facilitate domain experts in further analysis and identification of system level issues.**

***Keywords*-Clustering; Minimum spanning tree; Outlier detection; Sequential pattern mining;**

I. Introduction

Outlier detection has been studied and used to detect anomalous behavior in different domains. Outliers refer to data points that are significantly different from the rest of populations. They can happen due to mechanical issues, fraudulent behavior, human error and if they are not identified may lead to uncontrollable situations. Outlier detection refers to the problem of finding unusual patterns in data or unknown behaviors in a system [1], [2].

In this paper, we deal with outlier detection in sequence datasets. A sequence dataset is a collection of sequences of events or elements listed, often with concrete notion of time [3]. Due to importance of the sequential ordering of the events, sequential pattern mining for finding interesting subsequences in sequence datasets was introduced in 1995 [4]. Sequential pattern mining has a broad application in different fields such as bio-informatics [5], [6], marketing [7], network security [8], telecommunication [9], [10], and text mining [11], [12]. Data in these domains is highly dimensional and sparse which makes the identification of outliers more complex [13].

In this study, an outlier is defined as a small cluster (with respect to the number of matched sequences in a sequence dataset) which is significantly different from most of the frequent sequential patterns.

We propose an unsupervised 3-step approach that applies sequential pattern mining, cluster analysis, and a minimum spanning tree (MST) algorithm on a sequence dataset. In the first step, sequential pattern mining is used to extract frequent sequential patterns from data. In the second step, a clustering algorithm is applied on the extracted patterns in order to group the similar patterns together. Partitioning the patterns makes it possible in the third step to identify groups of patterns as outliers rather than detecting outliers individually. Consequently, this can lead to time complexity reduction in the proposed approach. In the third step, similar to [14], a minimum spanning tree is built on the clustering solutions in order to find clusters of outliers. By removing the longest edge(s) of the MST, the tree will be transformed to a forest. The small sub-tree(s) with few number of clusters (nodes) and/or with smaller sized clusters can be identified as outliers. The initial assumption is: the sub-trees with fewer nodes and smaller size contain patterns that happen rarely. Therefore, the clusters in these sub-trees are small, far and different from the clusters in the bigger sub-trees. The process of removing the longest edge(s) of the MST can also be performed by considering a user-specified threshold. The detected clusters of outliers can supply domain experts with a better understanding of the system behavior and facilitate them in the further analysis by mapping the detected patterns to the corresponding sequences. The proposed approach has been evaluated on smart meter data and video session data. The results of the evaluation on video session data has been discussed with the domain experts.

II. Background and Related Work

Outlier detection techniques have been studied and successfully applied in different domains. There exists a considerable number of studies that provide a comprehensive, and structured overview of the state-of-the-art methods and applications for outlier detection [1], [2], [15], [16]. This attention shows the importance of outlier detection techniques in different domains and the fact that they are domain specific.

Outlier detection techniques can be classified into three groups based on the availability of the labeled data [1], [2]: **1)** In the absence of prior knowledge of the data, unsupervised learning methods are used to determine outliers. The initial

978-1-5386-6805-4/18/$31.00 ©2018 IEEE
DOI 10.1109/ICMLA.2018.00182

assumption is that normal data represents a significant portion of the data and is distinguishable from faults or error; **2)** In the presence of labeled data, both normality and abnormality are modeled. This approach refers to supervised learning; **3)** Define what is normal and only model normality. This approach is known as semi-supervised learning since the algorithm is trained by labeled normal data, however, it is able to detect outliers or abnormalities. Semi-supervised outlier detections are more widely used compared to supervised techniques due to an imbalanced number of normal and abnormal labeled data.

The most common unsupervised learning method is cluster analysis, which is used for exploratory data analysis and grouping similar data into clusters. There are several clustering algorithms capable of detecting noise and eliminating it from the clustering solution such as DBSCAN [17], CRUE [18], ROCK [19], and SNN [20]. Even though such techniques can be used to detect outliers, the main aim of the clustering algorithm is to perform the partitioning task rather than identifying outliers.

This led to proposing clustering-based techniques that are capable of detecting: **1)** single-point outliers such as the application of Self Organizing Maps for clustering normal samples and identifying anomalous samples [21], and Expectation Maximization [22] for identifying the performance problems in distributed systems or **2)** groups of outliers such as the intrusion detection proposed by [23].

The application of MST has been studied by researchers in different fields including cluster analysis and outlier detection [14], [24]–[27]. A two-phase clustering algorithm is introduced for detecting outliers by [14]. In the first phase, a modified version of the k-means algorithm is used for partitioning the data. The modified k-means creates $k + i$ clusters, i.e., if a new data point is far enough from all clusters (k, number of clusters defined by the user), it will be considered as a new cluster (the $(k + i)^{th}$ cluster where, $i > 0$). In the second phase, an MST is built where, the tree's nodes represent the center of each cluster and edges show the distance between nodes. In order to detect outliers, the longest edges of the tree are removed. The sub-trees with a few number of clusters and/or smaller clusters are selected as outliers.

Wang et al. [24] developed an outlier detection by modifying k-means for constructing a spanning tree similar to an MST. The longest edges of the tree are removed to form the clusters. The small clusters are regarded as potential ouliters and ranked by calculating a density-based outlying factor.

A spatio-temporal outlier detection for early detection of critical events such as flood through monitoring a set of meteorological stations is introduced in [27]. Using geographical information of the data, a Delaunay triangulation network of the stations is created. The following step limits the connection of nodes to their closest neighbors while preventing far nodes from being linked directly. In the next step, an MST is constructed out of the created graph. In the final step, the outliers are detected by applying two statistical methods to detect exactly one or multiple outliers.

In this study, we propose a 3-step outlier detection approach that is specifically developed for sequence datasets. In the first step, frequent sequential patterns are extracted. In the second step, the selected patterns are clustered using the affinity propagation (AP) algorithm. In the third step, a minimum spanning tree similar to the study of Jiang et al. [14] is used to identify small groups of clusters as outliers.

III. Methods and Techniques

A. Sequential Pattern Mining

Sequential pattern mining is the process of finding frequently occurring patterns in a sequence dataset. The records of the sequence dataset contain sequences of events that often have chronological order. As examples of sequence data we can refer to customer shopping sequences, biological sequences, and video session events sequences.

Let $\mathcal{I} = \{i_1, i_2, ..., i_m\}$ be a set of all items. A sequence α defined as $\langle a_1, a_2, ..., a_j, ..., a_m \rangle$, where a_j is an itemset. Each itemset a_j is a subset of $\mathcal{I}$ that its items happened at the same time. In this study, each itemset (a_j) is a singleton. A sequence $\alpha = \langle a_1, a_2, ..., a_m \rangle$ is a subsequence of $\beta = \langle b_1, b_2, ..., b_n \rangle$ if and only if there exist integers $1 \leq k_1 < k_2 < ... < k_m \leq n$ and $a_1 \subseteq b_{k_1}, a_2 \subseteq b_{k_2}, ..., a_m \subseteq b_{k_m}$ [4]. Given a sequence dataset $\mathcal{T} = \{s_1, s_2, ..., s_j, ..., s_n\}$, where s_j is a sequence of itemsets, the support for α is the number of sequences that contain α as a subsequence. A sequence α is called a frequent sequential pattern if its support is equal or greater than user-specified support threshold.

To extract frequent sequential patterns the PrefixSpan algorithm [28] is used. PrefixSpan applies a *prefix-projection* method to find sequential patterns. Given a sequence dataset $\mathcal{T}$ and a user-specified threshold, the dataset is first scanned in order to identify all frequent items in sequences. All of these frequent items are considered as length-1 sequential pattern. After that, the search space is divided into a number of subsets based on the extracted prefixes. At last, for each subset a corresponding *projected dataset* is created and mined recursively.

Pei et al. [29] show in their study that PrefixSpan has the best overall performance compared to other sequential pattern mining algorithms such as GSP [30] and SPADE [31]. Therefore, the PrefixSpan algorithm is used for extracting sequential patterns in this study.

B. Similarity Measure

In order to calculate the similarity between the frequent sequential patterns with different lengths, we use *Levenshtein distance (LD)* metric [32]. The LD, also known as edit distance, is a string similarity metric that measures the minimum number of editing operations (insertion, deletion and substitution) required to change one string into the other. We used normalized LD, i.e., the result of each comparison is normalized by dividing it with the length of the longest pattern. The score ranges between 0 and 1. Score 0 implies 100% similarity between the two patterns and 1 represents no similarity. LD is a simple algorithm capable of measuring

the similarity between patterns with different lengths. In this study since the extracted patterns can have different length we choose to use LD as a similarity measure.

C. Clustering Method

The extracted patterns are clustered by using affinity propagation algorithm [33]. AP works based on the concept of exchanging messages between data points until a good set of exemplars (the most representative of a cluster) and corresponding clustering solution appears. The exchanged messages at each step assists AP to choose the best samples as exemplars and which data points should choose those samples to be their exemplars. AP adapts the number of clusters based on the data. However, the number of clusters can be controlled by *preference* parameter. That is, a high value of the preference will cause AP to form many clusters, while a low value will lead to a small number of clusters. If the preference value is not provided the median similarity or the minimum similarity will be used.

Unlike most clustering algorithms such as k-means that requires the number of clusters as an input, AP is able to estimate the number of clusters based on the data provided. AP can create clusters of different shapes and sizes, and the exemplars (the selected data points) are the representative of the clusters [34]. These characteristics make AP a suitable clustering algorithm for the chosen datasets in this study.

D. The Proposed Approach

The proposed approach consists of a preprocessing step (*Data segmentation*) and 3 main steps: 1) *Sequential patterns mining*, 2) *Frequent sequential pattern clustering*, and 3) *MST building and outlier detection analysis* as follows:

0) ***Data segmentation.*** The data is first partitioned into equal-sized segments in order to identify sequential patterns. Due to availability of daily patterns in the data, similar segments of similar days can be compared. In this paper, we have studied a daily segment.
1) ***Sequential patterns mining.*** The first step concerns the extraction of frequent sequential patterns and mapping them with records of a sequence dataset.
    a) ***Frequent sequential patterns finding.*** The PrefixSpan algorithm is used to find frequent sequential patterns from each segment. The extracted patterns can lead us to find sequences of events that based on their occurrence together assumed to be anomalous, also known as *collective* outliers [1]. Those sequential patterns that satisfy the user-specified support threshold will be stored as frequent patterns. Note that in sequential patterns the order of the events is important and by using the sequential pattern mining both the frequency and the order of events in the extracted patterns are taken into account.
    b) ***Frequent sequential patterns mapping.*** In the second step the extracted patterns are mapped with the source they come from and stored in a *selected_patterns* list. This relates to identifying those video sessions that contain the patterns or a device that the patterns are extracted from. The following step can lead us to find additional information about patterns such as pattern frequency and its occurrence time. The latter is also useful for finding a *contextual* or *conditional* outlier, i.e., a data point assumed to be anomalous only in a specific context [1].
2) ***Frequent sequential pattern clustering.*** Using the Levenshtein distance measure, as explained in sub-section *B*, the pairwise similarities between all patterns are calculated and the similarity matrix are constructed. The selected patterns are partitioned by applying affinity propagation on the created similarity matrix. Note, an advantage of using AP is that it can estimate the number of clusters from data.
3) ***MST building and outlier detecting.*** The third step includes two sub-steps concerning the construction of an MST, and the identification of sub-trees with the smallest size as outliers.
    a) ***MST building.*** The exemplars of the clusters are used for building a complete weighted graph where vertices of the graph are exemplars of the clusters and edges are the distance between them (traversing weight). Using the MST algorithm, the aim is to determine a sub-set of edges that connect all the vertices together without any cycles that has the minimum total edge weight.
    b) ***Outlier detection analysis.***
        i) The longest edge of the tree is removed. Note that there can be more than one edge to cut.
        ii) The constructed MST will be replaced by the created sub-trees, i.e., a single tree becomes a forest. Note that step (ii) can be repeated until the distance between nodes of the sub-trees become less or equal to a user-specified threshold. For example the threshold can be set to 0.5.
        iii) The sub-trees are ranked from smallest to largest based on the number of items they match within the sequence dataset. Following the definition of outliers that refers to patterns that happen rarely and sufficiently far away from other patterns [1], here the smallest sub-trees can be regarded as outliers.

IV. Experimental Methodology

A. Datasets

The proposed approach is evaluated on two datasets namely, *smart meter* and *video session*. The first dataset contains smart meters recorded events data provided by Elektro Ljubljana, a power distribution company in Slovenia [35]. The provided data contains 10,739,273 records that logged different events generated by 117,944 smart meters between May and August

2017. Due to the high number of data, we only considered data from May 2017 and sampled 30 devices out of 85,776 without replacement. To decrease the chance of any bias two datasets are created through sampling and the experiment ran on each set separately. This led to selection of 28 unique devices for each dataset, i.e., in total 56 distinct devices and 2 identical devices are sampled. The event sequences generated by the sampled devices for datasets 1 and 2 contain 40 and 44 unique event types respectively. The datasets together contain 36 identical event types. Each of these event types have an informative description and a unique ID that explain the status of a device at a specific time, e.g., '*Voltage OK L1*', '*Power down L2*' and '*Power up*'.

Table I summarizes detailed information about the smart meters dataset. Since daily activity of similar devices are monitored for a period of one month (May 2017), sequential patterns are extracted individually from devices in each daily segment. Therefore, the PrefixSpan user-specified support threshold (Step 1-a of the proposed approach) is set to be 1. Furthermore, to reduce the time complexity, only patterns with length between 2 to 7 are considered. On the other hand, if the interest is to know what kind of issues are mostly common between all devices, frequent patterns can be extracted from each segment and by considering all devices. That is, the PrefixSpan user-specified support threshold can set to be the minimum percentage that extracted patterns should appear in a segment.

TABLE I: Summary of the smart meter data, May 2017

No. of devices		85,776
No. of recorded event logs		2,265,08
No of Event types		135
	Sampled dataset 1	Sampled dataset 2
No. of selected devices	30 (28)	30 (28)
No. of recorded event logs	28,369	54,555
No. of event types	40 (4)	44 (8)

Note. The distinct number of devices or events in each dataset are listed in the parentheses.

The second dataset contains one month (February 2018) of video session data. The data is obtained from a large European telecommunication company and contains 288,669 unique video session IDs, 4,983,090 events, 19 event types and 23,485 video programs. Examples of the event types in video session data are '*Playback.Aborted*', '*Playback.BitrateChanged*', and '*Playback.PlayerReady*'.

Table II summarizes detailed information regarding this dataset. Since viewers receive a unique video session ID each time they login into their accounts, we extract frequent patterns that are common between all sessions in each daily segment. For this purpose after having some preliminary tests and discussions with the experts the user-specified support threshold is set to be 20%, i.e., the extracted patterns should at least appear in $(length\_of\_segment * 0.2)$ of the sessions. Moreover, we assume that the extracted patterns must contain at least 3 event types.

TABLE II: Summary of the video session data, February 2018

No. of video session IDs	288,669
No. of events	4,983,090
No. of video IDs	23,485
No. of event types	19

In all datasets, each event type represents an item. In the smart meter data, event sequences represent an overall status of each device per day while in the video session data, event sequences contain the quality related events and actions that have been performed by the viewers during the sessions. Moreover, event sequences in both datasets contain itemsets with exactly one event in each, i.e., the itemsets in this study are singletons.

B. Implementation and Availability

The proposed approach is implemented in Python version 3.6. The Python implementations of PrefixSpan and LD measure are fetched from [36] and [37], respectively. The affinity propagation algorithm is adopted from the scikit-learn module [38]. For constructing, manipulating, and visualizing a minimum spanning tree the NetworkX package is used [39]. The NetworkX package uses Kruskal's algorithm [40] for constructing the MST. The implemented code and the experimental results are available at GitHub[1].

V. Results and Discussion

A. Smart meter dataset

We performed random sampling on the smart meter dataset and created two datasets with 30 devices in each. More details can be found in *Datasets* sub-section and in Table I.

Applying the proposed approach on the first sampled dataset, leads to extracting 6,550 patterns. Using AP the collected patterns are partitioned into 253 clusters. Finally, by building the MST on top of the exemplars of the clustering solution and cutting the longest edge(s) three sub-trees are constructed. The two smallest sub-trees are identified as outliers. The patterns in these sub-trees are matched with 8 devices. Examples of detected patterns as outliers in daily operation of smart meters in May 2017 are as follows: {'Adjust time/date (old time/date)', 'Adjust time/date (new time/date)'}, {'Power down L3', 'Power down L1', 'Power down L2', 'Power restored L3', 'Power restored L1', 'Power restored L2'}, and {'Remote communication module OK', 'Communication board access error, PLC or GSM/GPRS module'}.

For the second sampled dataset, in total 6,676 patterns are identified. The extracted patterns are partitioned in 211 clusters and removing the longest edge(s) of the MST led to identifying 2 sub-trees out of 3 as outliers. Fig. 1 (Top) shows the constructed MST, the created forest after cutting the longest edges (edges A and B) of the MST, and the detected sub-trees as outliers. The patterns in these two clusters are matched with 10 devices. Examples of the identified

[1]https://github.com/shahrooz-abghari/MST-Clustering-Approach

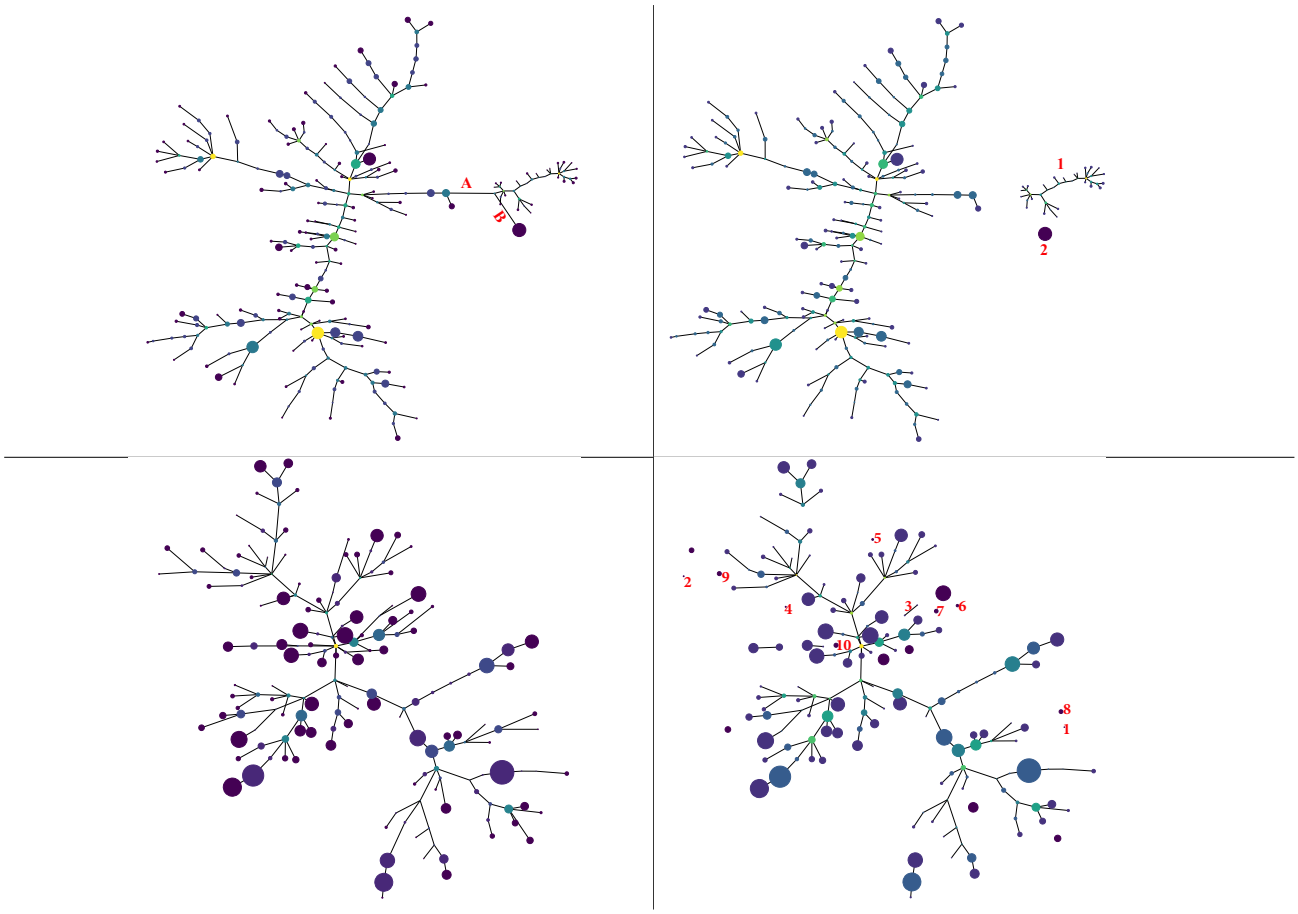


Fig. 1: **(Top-left)** The constructed MST before removing the longest edges on smart meter sampled dataset 1. Edges A and B represent the longest edges of the tree. **(Top-right)** The transformation of the constructed MST into a forest with 3 sub-trees after the longest edges are removed. The sub-trees 1 and 2 are considered as outliers based on their size. **(Bottom-left)** The constructed MST before removing the longest edges on video session dataset. **(Bottom-right)** The transformation of the constructed MST into a forest with 22 sub-trees after the longest edges are removed. The sub-trees are ranked from smallest to largest based on their size. The top 10 smallest sub-trees are considered as outliers. ***Note.*** The size of a node represents the number of smart meters or video sessions that are matched with it. The color of a node shows the degree of the node and is used only for the visualization purposes. The distance between edges range between [0,1].

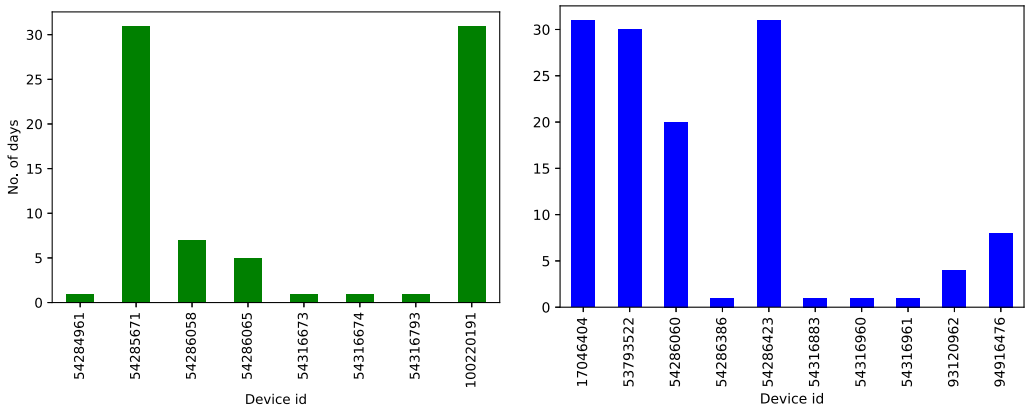
interesting sequential patterns are as follows: {'Wrong phase sequence', 'Wrong phase sequence'}, {'Under voltage L2', 'No under voltage L2 anymore'}, and {'Communication error with FLEX meter/Measuring system access error', 'Meter communication OK with FLEX meter/Measurement System OK', 'Communication error with FLEX meter/Measuring system access error'}. Table III summarizes the results of the experiment on the smart meter data.

The results of the experiment on the smart meter data showed that the detected patterns as outliers are matched with 8 devices in the sampled dataset 1 and 10 devices in the sampled dataset 2. Fig. 2 shows the identified device ids with issues and the number of days they were faulty. Table IV presents the top 5 sequential patterns that identified as outliers for each dataset. Perhaps only some of these patterns represent serious issues and some only relate to miss-configuration such as Adjust time/date. However, since such sequences of patterns occurred rarely in a normal daily activity of the monitored devices they have been detected as outliers. Nevertheless, further analysis and domain experts' opinion are needed to determine the severity of the issues raised by these patterns.

TABLE III: The results of the experiment for smart meter data

Dataset 1	No. of extracted patterns	6,550
	No. of patterns detected as outlier	241
	No. of clusters	253
	No. of detected outliers	1 sub-tree with 1 node 1 sub-tree with 39 nodes
	No. of devices with issue	6
Dataset 2	No. of extracted patterns	6,676
	No. of patterns detected as outlier	215
	No. of clusters	211
	No. of detected outliers	2 sub-trees, 1 node each
	No. of devices with issue	10

Note. Numbers inside the parentheses represent unique devices.



(a) In dataset 1, 8 devices are identified with issues. (b) In dataset 2, 10 devices are identified with issues.

Fig. 2: Identified smart meter with issues for both sampled datasets 1 and 2, May 2017.

TABLE IV: Top 5 sequential patterns identified as outliers for smart meter sampled datasets, May 2017

	Pattern	Freq. of the pattern
Dataset 1	$\langle 165, 79, 165, 79, 79, 165, 165\rangle$	27
	$\langle 10, 11, 11, 10, 11, 10, 11\rangle$	15
	$\langle 20, 22\rangle$	3
	$\langle 21, 20\rangle$	3
	$\langle 397, 396, 393, 395, 392, 63, 346\rangle$	1
Dataset 2	$\langle 20, 20\rangle$	32
	$\langle 245, 245, 245, 245, 245, 245, 245\rangle$	31
	$\langle 63, 63, 63, 63, 63, 63, 63\rangle$	28
	$\langle 21, 21\rangle$	17
	$\langle 22, 184\rangle$	17

Note. Event names equivalent to each ID are as follows: **ID** $= 10$, *Adjust time/date (old time/date)*, **ID** $= 11$, *Adjust time/date (new time/date)*, **ID** $= 20$, *Over voltage on L1*, **ID** $= 21$, *Over voltage on L2*, **ID** $= 22$, *Over voltage L3*, **ID** $= 63$, *Wrong phase sequence*, **ID** $= 79$, *Communication board access error, PLC or GSM/GPRS module*, **ID** $= 165$, *Remote communication module OK*, **ID** $= 184$, *No over voltage L3 anymore*, **ID** $= 245$, *E Meter command error*, **ID** $= 346$, *Voltage L2 normal*, **ID** $= 392$, *Power down L1*, **ID** $= 393$, *Power down L2*, **ID** $= 394$, *Power down L3*, **ID** $= 395$, *Power restored L1*, **ID** $= 396$, *Power restored L2*, **ID** $= 397$, *Power restored L3*.

B. Video session dataset

We applied our proposed approach on one month (February 2018) video session data. In total 1,493 sequential patterns are mined. AP partitioned the patterns into 170 clusters and cutting the longest edges of the MST led to 22 sub-trees. We sort the sub-trees based on the number of video sessions they matched with, and choose the top 10 smallest sub-trees as outliers. Fig. 1 (Bottom) shows the constructed MST, the created forest after cutting the longest edges of the MST, and the top 10 smallest sub-trees identified as outliers. In total 10,121 video sessions are matched with patterns in these sub-trees. Examples of identified pattern as outlier are {'Playback.BufferingStarted', 'Playback.ScrubbedTo', 'Playback.Aborted'} and {'Playback.BitrateChanged', 'Playback.Resumed', 'Playback.Completed'}. Table V summarizes

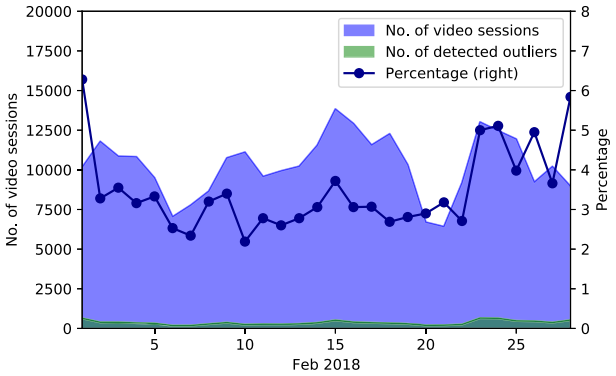


Fig. 3: Visualization of the total number of video sessions vs. detected outliers for each day of February 2018 together with the percentage of the outliers.

the results of the experiment on the video session data.

The results of the experiment on the video session dataset showed on average 3.5% of the sessions in each day contained outliers. Fig. 3 shows the total number of video sessions against the number of detected sessions as outliers and their percentage. In days 1, 23, 24, 26, and 28 of February higher number of outliers are detected. In the video session data, it is

TABLE V: The results of the experiment for video session data

No. of extracted patterns	1,493
No. of patterns detected as outliers	88
No. of clusters	170
No. of sub-trees	22
No. of detected outliers	1 sub-tree with 2 nodes 9 sub-trees, 1 node each
No. of video sessions with issue	10,121

hard to draw any conclusions regarding the detected patterns as

outliers without experts' validation. Unlike smart meter data that each event shows explicitly the status of a device, the event types in video session data are more general and most events can appear in both sessions with *good* and *bad* quality. However, the ratio of quality related events such as *Playback.BitrateChanged*, $ID = 2$, and *Playback.BufferingStarted*, $ID = 4$ or *Playback.BufferingStopped*, $ID = 5$ in a viewer's session can assist us to judge the quality of the session. A sudden increase in any of these three events in a session and simultaneously for many viewers can represent an issue at the system level.

Table VI shows the top 10 sequential patterns that identified as outliers and the number of video sessions that matched with them. Among these ten patterns seven of them (patterns 1-6 and 9) contain the quality related events that are mentioned earlier. On the other hand, there are two patterns (7 and 9) that contain an event type *Playback.BufferingEnded*, $ID = 3$. This event often generates when the viewers scrub the video. Scrubbing is an action that helps a viewer to navigate through a video program to watch from a specific section. However, sometimes the viewers have to scrub the video due to a frozen screen. Nevertheless, if the ratio of *Playback.BufferingEnded* increases inside a video session and at the same time for many sessions can relate to an issue at the system level. We have

TABLE VI: Top 10 sequential patterns identified as outliers for video session dataset, February 2018

No.	Pattern	Freq. of the pattern
1	$\langle \mathbf{2}, 16, 6 \rangle$	1613
2	$\langle 11, 7, \mathbf{2}, 18 \rangle$	838
3	$\langle 11, 17, \mathbf{4} \rangle$	653
4	$\langle 11, 7, \mathbf{2}, 16 \rangle$	477
5	$\langle 12, \mathbf{5}, 1 \rangle$	459
6	$\langle \mathbf{4}, 17, 1 \rangle$	445
7	$\langle 11, 3, 15 \rangle$	403
8	$\langle 18, 17, 1 \rangle$	401
9	$\langle 11, 3, \mathbf{4} \rangle$	298
10	$\langle 12, 14, 1 \rangle$	290

Note. The numbers in bold represent the quality related events. Event names equivalent to each ID are as follows: $\mathbf{ID} = 1$, *Playback.Aborted*, $\mathbf{ID} = 2$, *Playback.BitrateChanged*, $\mathbf{ID} = 3$, *Playback.BufferingEnded*, $\mathbf{ID} = 4$, *Playback.BufferingStarted*, $\mathbf{ID} = 5$, *Playback.BufferingStopped*, $\mathbf{ID} = 6$, *Playback.Completed*, $\mathbf{ID} = 7$, *Playback.Created*, $\mathbf{ID} = 11$, *Playback.HandshakeStarted*, $\mathbf{ID} = 12$, *Playback.InitCompleted*, $\mathbf{ID} = 14$, *Playback.PlayReady*, $\mathbf{ID} = 15$, *Playback.PlayerReady*, $\mathbf{ID} = 16$, *Playback.Resumed*, $\mathbf{ID} = 17$, *Playback.ScrubbedTo*, $\mathbf{ID} = 18$, *Playback.Started*.

discussed the obtained results with domain experts. In order to validate the results, the experts asked us to randomly select 18 video sessions (9 normal and 9 abnormal) from February 1, 2018. The labels of the video sessions were unknown for the experts. The validation shows only 3 video sessions can be considered as true anomalies and the other sessions are normal. This means $12/18 = 67\%$ of the video sessions are labeled correctly by the proposed approach. Further analysis of the experts' comments has revealed that assessing the quality of a video session is not an easy task and sometimes can be subjective. For example, the latter is supported by the experts' comments concerning the following 4 video sessions (V_1 to V_4) out of 6 that are mislabeled as outliers by the proposed approach:

- $\mathbf{V_1}$. "Short session (15,5 seec), no buffering, good bitrate, hard to tell, I tend to OK."
- $\mathbf{V_2}$. "No buffering, rather short, 10 sec, bitrate rather good. OK."
- $\mathbf{V_3}$. "Probably still OK considering the duration of the playback (1055 sec), but some buffering (35 sec in total), with varying bitrate, I tend to OK."
- $\mathbf{V_4}$. "Some buffering, but below 0,5% of the play duration, played 612 sec, error event at the end of the session with no further explanation, played 10 minutes with good bitrate, I tend to OK."

The validation of the results has shown that the proposed approach is able to identify video sessions that are significantly different from the majority of the sessions due to occurrence of some specific event types. The identified anomalous sequential patterns can help the domain experts to understand the outlying properties of the detected outliers.

VI. Conclusion

In this study, we have presented an outlier detection for sequence datasets. Our approach combines sequential pattern mining, clustering, and minimum spanning tree to identify outliers. We have shown that the proposed approach can facilitate the domain experts in identification of outliers. Building the minimum spanning tree on top the clustering solution can lead to identifying clusters of outliers. This can reduce the time complexity of the proposed approach. Moreover, in this study we have looked into collective outliers, sequences of events that based on their occurrence together assumed to be anomalous, which may help to find the outlying properties of the detected outliers.

The proposed approach has been applied on two sequence datasets, smart meter data and video session data. Both datasets contain sequences of event types that either shows the operational status of a smart meter or the current action that takes place in a viewer's video session. The results of the experiments on the smart meters data are more comprehensible compared to the video session data. The main reason is the fact that the event types in smart meters are explicitly detailed, explaining the status of the devices. However, in video session data the event types are general which requires more investigation and experts' knowledge in order to detect video sessions with quality issues. The validation of the results on video session data by the domain experts showed that 67% of the labeled sessions by the proposed approach were correct.

For future work, we are going to further evaluate and validate the proposed approach by using different distance measures and clustering algorithms. These two parameters may have a strong correlation and it worths to further study how such correlation can affect the final results.

In this study, the MST is constructed from a complete weighted graph using the exemplars of the clusters. However, according to Cipolla et al.'s study a Delaunay triangulation

network can be used to create a simplified graph. The Delaunay triangulation limits the number of connections of a node to its closest neighbors. This makes the constructed MST from this simplified graph a better representative compared to the complete graph. Therefore, we are also interested in testing whether the Delaunay triangulation network can be integrated into our approach.

ACKNOWLEDGMENT

This work is part of the research project "*Scalable resource-efficient systems for big data analytics*" funded by the Knowledge Foundation (grant: 20140032) in Sweden.

REFERENCES

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, p. 15, 2009.

[2] V. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial Intelligence Review*, vol. 22, no. 2, pp. 85–126, 2004.

[3] J. Han, H. Cheng, D. Xin, and X. Yan, "Frequent pattern mining: Current status and future directions," *Data Mining and Knowledge Discovery*, vol. 15, no. 1, pp. 55–86, 2007.

[4] R. Agrawal and R. Srikant, "Mining sequential patterns," in *Proc. of the 11th Int'l Conf. on Data Engineering*. IEEE, 1995, pp. 3–14.

[5] T. P. Exarchos, C. Papaloukas, C. Lampros, and D. I. Fotiadis, "Mining sequential patterns for protein fold recognition," *J. of Biomedical Informatics*, vol. 41, no. 1, pp. 165–179, 2008.

[6] J. Guan, D. Liu, and D. A. Bell, "Discovering motifs in dna sequences," *Fundamenta Informaticae*, vol. 59, no. 2-3, pp. 119–134, 2004.

[7] Y.-L. Chen, M.-H. Kuo, S.-Y. Wu, and K. Tang, "Discovering recency, frequency, and monetary (rfm) sequential patterns from customers purchasing data," *Electronic Commerce Research and Applications*, vol. 8, no. 5, pp. 241–251, 2009.

[8] L.-C. Wuu, C.-H. Hung, and S.-F. Chen, "Building intrusion pattern miner for snort network intrusion detection system," *J. of Systems and Software*, vol. 80, no. 10, pp. 1699–1715, 2007.

[9] F. Eichinger, D. D. Nauck, and F. Klawonn, "Sequence mining for customer behaviour predictions in telecommunications," in *ECML/PKDD 2006 Workshop on Practical Data Mining: Applications, Experiences and Challenges*, 2006, pp. 3–10.

[10] T. H. N. Vu, K. H. Ryu, and N. Park, "A method for predicting future location of mobile user for location-based services system," *Computers & Industrial Engineering*, vol. 57, no. 1, pp. 91–105, 2009.

[11] S. Jaillet, A. Laurent, and M. Teisseire, "Sequential patterns for text categorization," *Intelligent Data Analysis*, vol. 10, no. 3, pp. 199–214, 2006.

[12] P. C. Wong, W. Cowley, H. Foote, E. Jurrus, and J. Thomas, "Visualizing sequential patterns for text mining," in *Symp. on Information Visualization*. IEEE, 2000, pp. 105–111.

[13] C. C. Aggarwal and P. S. Yu, "Outlier detection for high dimensional data," in *ACM Sigmod Record*, vol. 30, no. 2. ACM, 2001, pp. 37–46.

[14] M.-F. Jiang, S.-S. Tseng, and C.-M. Su, "Two-phase clustering process for outliers detection," *Pattern Recognition Letters*, vol. 22, no. 6, pp. 691–700, 2001.

[15] Y. Zhang, N. Meratnia, and P. Havinga, "Outlier detection techniques for wireless sensor networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 12, no. 2, pp. 159–170, 2010.

[16] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han, "Outlier detection for temporal data: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 9, pp. 2250–2267, 2014.

[17] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *KDD*, vol. 96, no. 34, 1996, pp. 226–231.

[18] S. Guha, R. Rastogi, and K. Shim, "Cure: An efficient clustering algorithm for large databases," in *ACM Sigmod Record*, vol. 27, no. 2. ACM, 1998, pp. 73–84.

[19] ——, "Rock: A robust clustering algorithm for categorical attributes," in *Proc. of the 15th Int'l Conf. on Data Engineering*. IEEE, 1999, pp. 512–521.

[20] L. Ertöz, M. Steinbach, and V. Kumar, "Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data," in *Proc. of the 2003 SIAM Int'l Conf. on Data Mining*. SIAM, 2003, pp. 47–58.

[21] F. A. González and D. Dasgupta, "Anomaly detection using real-valued negative selection," *Genetic Programming and Evolvable Machines*, vol. 4, no. 4, pp. 383–403, 2003.

[22] X. Pan, J. Tan, S. Kavulya, R. Gandhi, and P. Narasimhan, "Ganesha: Blackbox diagnosis of mapreduce systems," *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 3, pp. 8–13, 2010.

[23] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, "A geometric framework for unsupervised anomaly detection," in *Applications of data mining in computer security*. Springer, 2002, pp. 77–101.

[24] X. Wang, X. L. Wang, and D. M. Wilkes, "A minimum spanning tree-inspired clustering-based outlier detection technique," in *Ind. Conf. on Data Mining*. Springer, 2012, pp. 209–223.

[25] A. C. Müller, S. Nowozin, and C. H. Lampert, "Information theoretic clustering using minimum spanning trees," in *Joint DAGM (German Association for Pattern Recognition) and OAGM Symp.* Springer, 2012, pp. 205–215.

[26] G.-W. Wang, C.-X. Zhang, and J. Zhuang, "Clustering with prims sequential representation of minimum spanning tree," *Applied Mathematics and Computation*, vol. 247, pp. 521–534, 2014.

[27] E. Cipolla, U. Maniscalco, R. Rizzo, D. Stabile, and F. Vella, "Analysis and visualization of meteorological emergencies," *Ambient Intelligence and Humanized Computing*, vol. 8, no. 1, pp. 57–68, 2017.

[28] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu, "Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth," in *Proc. of the 17th Int'l Conf. on Data Engineering*, 2001, pp. 215–224.

[29] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu, "Mining sequential patterns by pattern-growth: The PrefixSpan approach," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 11, pp. 1424–1440, 2004.

[30] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," *Advances in Database Technology–EDBT'96*, pp. 1–17, 1996.

[31] M. J. Zaki, "SPADE: An efficient algorithm for mining frequent sequences," *Machine Learning*, vol. 42, no. 1, pp. 31–60, 2001.

[32] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10, no. 8, 1966, pp. 707–710.

[33] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, 2007.

[34] U. Bodenhofer, A. Kothmeier, and S. Hochreiter, "Apcluster: An r package for affinity propagation clustering," *Bioinformatics*, vol. 27, no. 17, pp. 2463–2464, 2011.

[35] Elektro Ljubljana, "Smart meters recorded events dataset," 2018. [Online]. Available: https://data.edincubator.eu/organization/elektro-ljubljana-podjetje-zadistribucijo-elektricne-energije-d-d

[36] C. Gao, "Prefixspan algorithm source code," 2015. [Online]. Available: https://github.com/chuanconggao/PrefixSpan-py

[37] B. Jain, "Edit distance algorithm source code," -. [Online]. Available: https://www.geeksforgeeks.org/dynamic-programming-set-5-edit-distance

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[39] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using networkx," Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2008.

[40] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proc. of the American Mathematical Society*, vol. 7, no. 1, pp. 48–50, 1956.