On the use of online clustering for anomaly detection in trace streams

Renato Vertuam Neto
State University of Londrina (UEL)
Londrina, Brazil
renato.vertuam@uel.br

Gabriel Marques Tavares
Università degli Studi di Milano (UNIMI)
Milan, Italy
gabriel.tavares@unimi.it

Paolo Ceravolo
Università degli Studi di Milano (UNIMI)
Milan, Italy
paolo.ceravolo@unimi.it

Sylvio Barbon Jr.
State University of Londrina (UEL)
Londrina, Brazil
barbon@uel.br

ABSTRACT

Identifying anomalies in business processes is a challenge organizations face daily and are critical for their operations' data flow, whether public or private. Most current techniques face this challenge by requiring prior knowledge about business process models or specialists intervention to support the usage of state of the art methods, such as supervised machine learning. Also, the techniques tend to perform offline towards achieving consistent predictive results. In this work, we propose identifying the effectiveness of an online clustering method, particularly Autocloud. This algorithm is able to perform anomaly detection in trace streams meeting real-life requirements. Autocloud is an autonomous, evolutionary, recursive online clustering algorithm that requires little memory to provide insights from anomalous patterns in real-time. Moreover, this clustering algorithm does not require previous training or even prior knowledge from the application domain. Experiments were carried out with six process models, six different anomalies over 1,000, 5,000, and 10,000 event traces, generating a total of 630 datasets. The experiments confirmed the algorithm's ability to detect anomalies in those event traces, paving the way for more reliable information systems grounded on an automatic conformance checking of desirable business process execution.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; • **Learning paradigms** → **unsupervised learning**; • **Information systems** → **Data stream mining**.

KEYWORDS

Clustering algorithms, Process mining, Anomalies, Data mining, Data stream mining, Information systems, Information systems applications.

SBSI 2021, June 7–10, 2021, Uberlândia, Brazil

ACM ISBN 978-1-4503-8491-9/21/06…$15.00
https://doi.org/10.1145/3466933.3466979

ACM Reference Format:
Renato Vertuam Neto, Gabriel Marques Tavares, Paolo Ceravolo, and Sylvio Barbon Jr.. 2021. On the use of online clustering for anomaly detection in trace streams . In *XVII Brazilian Symposium on Information Systems (SBSI 2021), June 7–10, 2021, Uberlândia, Brazil.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3466933.3466979

1 INTRODUCTION

Information Systems (IS) regulate, control, and record the execution of events from business processes in organizations. Business event data are stored in the form of event logs, which is a process fingerprint able to provide robust management and verification of business processes. Event logs comprehend a wide range of information, such as executed activities sequences, time stamps, attributes dependencies, concurrent behavior, and others [1]. In the last decades, a research area called Process Mining (PM) has been devoted to developing techniques for extracting knowledge from the massive amount of event logs recorded by IS. More specifically, PM techniques are usually applied for discovering, monitoring, and enhancing real business processes [18]. PM methods ingest an *event log*, which is composed of unique events recording activities executed at a certain time and belonging to a specific process instance. In PM literature, a process instance is known as a case. *Activity*, *timestamp* and *case identifier* are obligatory attributes for all events being processed by PM algorithms. Moreover, it follows that a unique sequence of time-ordered events of the same case is called *trace*.

In addition to bringing more intelligence to business process analysis, PM gives IS the ability to react to various anomalies and deviations. The detection of abnormal process instances is of utmost interest for organizations as the detection of incorrect executions can avoid frauds and save resources [21, 29]. Traditionally, anomaly detection in PM is driven by conformance checking approaches, which aim at comparing traces and process models, detecting control and data-flow deviations [27]. Control-flow anomalies concern activities sequences within traces, i.e., activities executed in the wrong order or missing activities. Instead, data-flow anomalies relate to unexpected attribute values, such as a user managing an activity without proper access to it.

However, the classical PM approach for anomaly detection is designed for *offline* scenarios, analyzing historical batches of event logs, possibly with multiple passes over the data. From a business

perspective, this method is deficient as the real-time assessment of processes is crucial to manage resources and react to incorrect behavior in time [16]. Furthermore, *online* scenarios pose several additional challenges, such as memory management, possible infinite streams, concept drift, and the need for regular model updates. Recently, some approaches emerged as a solution for conformance checking in online scenarios [11, 31, 33]. In [11], the method converts a process model into a transition system annotated with arcs describing possible deviations from the model. This pre-computation allows real-time conformance checking assessment, but the requirements in terms of memory consumption are high, and no model update is discussed. The approach in [33] supports online conformance checking for incomplete cases but relies on the backtracking procedures required for alignments, known to be resource costly [19]. In [31], the authors propose a technique that creates and maintains an updated process model in online environments. However, the model is based on a histogram representation that does not preserve event order, an essential asset for online PM anomaly detection.

To overcome these issues, we combined word2vec [24], a Natural Language Processing (NLP) encoding technique, with Autocloud [6], an online clustering algorithm, to support the detection of anomalous behavior in trace streams. Autocloud is an evolutionary and recursive algorithm that is memory-friendly while maintaining a low response time. Moreover, it does not require previous training or even any preliminary knowledge about the trace stream.

To evaluate our approach's performance, we developed a set of 630 event logs based on six different models, six anomaly types and four levels of incidence. This way, the synthetic event logs represent a wide range of real-life scenarios experienced by organizations. Moreover, we further investigated Autocloud's capacity of detecting anomalies by a hyperparameter tuning process. We obtained promising results for the anomaly detection in online business processes by reaching an accuracy of 0.96 in specific configurations.

The remainder of this paper is organized as follows. Section 2 presents the literature on anomaly detection in online PM. Section 3 discusses the application of clustering algorithms to online environments. Section 4 details our proposed approach for anomaly detection in trace streams, and Section 5 presents the event logs used for the experiments along with the metrics to assess performance. We analyze the results and discuss their implications in Section 6. Finally, we conclude the paper in Section 7.

2 RELATED WORK

Since our work's scope is to detect anomalies in online PM using an NLP-inspired technique for trace encoding, we focus the literature review on online conformance checking and application of NLP techniques to business processes.

Traditional anomaly detection in PM is usually addressed by methods based on conformance checking [5, 7–9, 26]. The identification of anomalous instances is based on a comparison between log traces and a process model, i.e., traces that deviate from the standard model are interpreted as anomalies [32]. However, these approaches are limited to offline scenarios, where one has access to historical recordings of a business process. The limitation relies on the fact that often no measures can be taken against past anomalies. Hence, organizations are interested in the detection of anomalies on the fly, that is, in online environments, allowing for a quick reaction to anomalous behavior. Above that, online settings present additional constraints such as memory limitation, potentially unbounded stream size, and minimization of response latency [16].

In offline conformance checking, alignments are used to detect anomalies. These techniques compare traces to valid execution sequences determined by the model [15]. Building from that, in [33], the authors propose the computation of alignment-based metrics for online conformance checking. The approach extracts the prefix-alignments from partially executed traces as the last activity that might not have been reached. However, this relaxed version of alignments leverages the number of false negatives. Thus, lower recall values are expected. This approach is extended in [28] with the reduction of conformance checking to the shortest path problem. The authors model a synchronous product net, which is composed of a trace and a workflow net. The state-space of the synchronous product net is incrementally explored so new instances can benefit from the previously calculated results.

In [34], the authors propose a method based on probabilistic automata to identify and filter infrequent behavior in event streams. In this context, rare or noisy instances, independently of the root cause, are considered anomalies as they do not conform to the expected process behavior. The approach maintains the filtering mechanism incorporates a sliding window and each ingested event. The filter represents the process's behavior captured within the window and is built by an ensemble of probabilistic automata, capturing the underlying distribution of relations between activities. Therefore, for a new event, its correspondent instance state is assessed, and based on the probability distribution, a decision is made to discard or not the event.

A framework for online conformance checking is proposed in [12]. The method builds from the idea of unfolding Petri nets [23], which are obtained from the original business process model. From the unfolded Petri net, behavioral patterns capturing the relation between activities are extracted. This way, during the stream processing, the observed behaviors are compared with the previously extracted patterns, thus assessing the compliance between real and expected activity relations. However, the steps until the extraction of the process patterns are performed offline and imply in an already known process model, which is not the case in many real-life applications.

Regarding the application of NLP techniques for modeling business process behavior, few works have explored the intersection between these areas. In [17], the authors use the doc2vec (an extension of the well-known word2vec algorithm [24]) method to encode trace behavior. For this approach, traces are interpreted as paragraphs, and the encoding feature space captures the relations between different paragraphs. The authors from [22] used word2vec in combination with one-class classification algorithms to detect anomalies in business processes. In [29], the authors also use word2vec and compare it against traditional conformance metrics.

Our work lies at the intersection between NLP-based trace representations and online environments. For that, we investigate the application of word2vec along with an online clustering method for the detection of anomalous process instances.

3 ONLINE CLUSTERING ALGORITHMS

Data stream mining has become an area of research of interest in recent years. The use of clustering algorithms is an efficient method to overcome the problem of extracting the valuable knowledge held in data streams in real-time. It can be applied in the most diverse fields of activity, whether in financial market transactions, telephone company records, devices, IoT sensors, and analyze business processes in e-commerce and Enterprise Resource Planning (ERP).

Invariably in the literature, most of the methods proposed for this purpose consider that the algorithm has some advanced knowledge about the set of samples to be analyzed. This knowledge is presented with a mathematical model, which describes the data, thus helping the algorithm have this prior understanding of the data. Then, with the increased use of these online applications, there was also a need to analyze the data generated from them, commonly known as data streaming. This way, the search for online clustering algorithms for real-time analysis has increased [25]. CluStream [3], DenStream [14], StreamKM++ [2] are among the most notable algorithms in this area.

Regularly, these algorithms divide their processes into two phases, which are online and offline. In [20] the author categorizes these algorithms taking into account the form of the approach used to perform the grouping of data.

The authors [3], discuss in this article a way to approach the problem so that with the use of CluStream that divides the grouping process into two parts, online and offline, that is, with this division in two steps, the algorithm would be able to offer greater malleability for exploring the evolutionary nature of clusters. In its online phase, reduced storage of detailed statistical data would be carried out. In its offline phase, the process uses this stored summary information.

In DenStream [14] we are presented with a clustering algorithm in an evolving data stream. This algorithm and CluStream use two steps, in its online phase of maintenance of the created micro-cluster and its offline phase, and perform the final clusters' generation.

StreamKM++ [2], consists of a two-phase algorithm, similar to the previous ones (i.e., an online and an offline phase). Unlike DenStream, based on density, StreamKM++ is grounded on partitioning towards creating data structures commonly defined as trees. These trees represent a subset of input data. It uses blending and reduction techniques. Also, in two stages, a merging stage and a reduction stage.

Data clustering is the usual method for solving problems of different classes in different fields of activity. Especially in our explored issue, the detection of anomalies in traces of business processes in the data stream. Moreover, clustering algorithms do not require labels, which is a common constraint for other types of online algorithms.

The methods proposed in the literature require little or some advanced knowledge about the samples.

In this article, for the experiments of our method of grouping and detecting anomalies in data streams, we use Autocloud to process and cluster the traces from the data stream. Autocloud is entirely data-driven and does not require specific mathematical models or any prior knowledge of the data set to be analyzed. It is based on the recently introduced concept of Typicality and Eccentricity Data Analysis (TEDA) [4].

3.1 Autocloud

Autocloud [6] is an algorithm that uses data analysis based on typicality and eccentricity (TEDA) [4] represented by an alternative analytical structure. Within its purpose, each sample obtained through a data stream Autocloud determines whether or not it belongs to a Data Cloud.

A Data Cloud is a structure used by Autocloud to group the samples obtained. Unlike the other algorithms, Autocloud creates data clouds instead of clusters. The comparison between data clouds and clusters has been widely discussed in the literature when dealing with the clustering algorithm. Notably, data clouds have no predefined format or geometric limits. They are representations of all samples from the data stream.

For each sample received, Autocloud checks if the sample received has any similarity, eccentricity, and typicality with the current Data Cloud. In case of a minimum similarity between the sample and the Data Cloud, the latter is updated. On the other hand, if the sample shows eccentricity concerning the data cloud, a new data cloud will be created to represent this new sample. Data clouds seek through this process, and with the use of TEDA represent as accurately as possible the samples received from the data stream in their data clouds.

The eccentricity [4] reflects how distant a given sample is from the data cloud or other compared samples. A sample with high eccentricity in relation to the other samples is a strong candidate to be an anomaly.

The typicality [4] is used to form the data clouds like fuzzy pertinence functions, but it does not require previous knowledge. The typicality depicts the analysis of the sample's frequency and the spatial distribution of these data simultaneously.

4 PROPOSED APPROACH

The proposed approach could be split into Acquisition, Encoding, and Online Clustering. The acquisition step regards the conversion of event logs into traces. A trace is a set of events representing a business process instance. An event is a unique activity. Each event can contain one or more activities or attributes, and finally, a log is a set of traces. The second step, Encoding, allows performing online clustering over the feature vector representation of traces. Finally, the Online Clustering with Autocloud delivers a data cloud perspective of our traces supporting the anomalous trace detection. Figure 1 shows an overview of the proposed method.

4.1 Trace Acquisition

Our method consists of acquiring samples, events from a data stream, events that together form a trace. This trace represents a particular business process. And machine learning techniques cannot act directly on these event logs, so we need to somehow prepare the data so that they are prepared to apply these techniques [13, 30].

Events are evaluated from the moment they arrive through the data flow, treating each one individually for trace extraction. Our method implements a window to acquire events from the stream

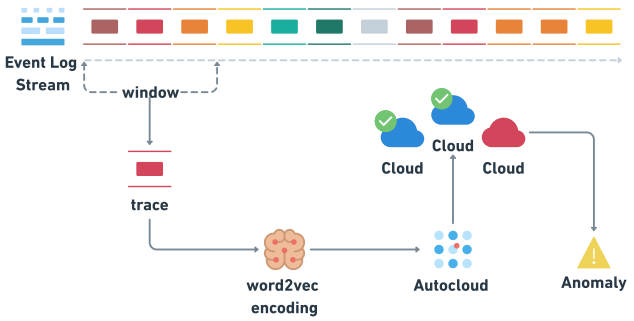

Figure 1: Proposed approach overview, from Event Log Stream to Anomaly Detection

and convert them into traces using a data stream monitoring system. A hyperparameter is used to define the number of events related to trace extraction, similarly to a sliding window. Considering a window of 100 events, we group all events from the same case keeping the corresponding sequence towards modeling all possible traces from this window. This particular hyperparameter is related to the evaluated domain, particularly regarding the average number of cases, its length ant parallel occurrence. It is necessary to mention that our proposal does not address event log streams but trace streams.

4.2 Encoding

To meet the sample preparation and execute this process, i.e., the encoding, we chose word2vec [24]. Word2vec is an encoding method grounded on a neural network to outline the linguistic context in a corpus. This process generates a vector with the weights learned through the learning that results from the use of the neural network [22].

Word2vec is a fundamental step of our approach because Autocloud cannot work directly with business process traces in their original representation. The efficiency of word2vec in the incorporation of words (e.g., process activities), motivated the choice of its as our encoding in our method. This way, word2vec converts the trace representation to a feature vector representation space, providing the suitable input for Autocloud. In other words, Autocloud processes a stream of encoded traces using word2vec representations. Next, Autocloud processes (i.e., online clustering) each encoded trace, by calculating the eccentricity and typicality. Using these metrics, the online clustering algorithm creates the data clouds representing a given event log state. Using the data cloud it is possible to have insights about anomalies, such as demonstrated in section 3.

For the process of encoding traces with word2vec, three crucial hyperparameters need to be defined: *size*, *window*, *min_count*. The *size* hyperparameter represents the dimension of the final vector after word2vec finishes its work. *Window* defines the maximum distance between the current and the predicted word in a sentence. *Min_count* defines that the algorithm ignores all words with a total frequency lower than value of this parameter. In our method, we set the following values for hyperparametrs: *size* = 50, *window* = 3 and *min_count* = 3, workers = 6.

At this point in the process, we started training the word2vec model by feeding it with data from the stream. After inserting the traces in word2vec for training, we took the sample from the algorithm's memory. When word2vec finishes the training with the sample received, it returns an array with the trace events' representation, encoded.

It is important to mention that other encoding methods could take place. Our suggestion is based on current literature and the descriptive performance of word2vec.

4.3 Clustering Algorithm - Autocloud

Widely discussed in the literature, the algorithms for data clustering are employed for solving problems involving different patterns and in various domains (data mining, image images, etc.) without a supervision influence in terms of machine learning. In our case, specifically for mining processes through the use of a data stream.

In most of the literature's proposals, it is possible to highlight the prior demand for knowledge about the analyzed datasets. Usually, this knowledge is represented through a mathematical model to characterize samples' behavior or even by using specialists to perform labeling tasks for supervised machine learning.

We propose to use Autocloud as the algorithm for data clustering. Similarly to unsupervised stream algorithms found in the literature, Autocloud does not need prior information about the samples of the data set to be processed as described in the section 3 [6]. As mentioned in the related work section, there are several other online clustering algorithms, but different from the others, Autocloud performs fast and requires a single hyperparameter. In comparison to DenStream, which has five hyperparameters, Autocloud presents a considerable advantage supporting our claim to reduce previous knowledge demand. The tuning strategies related to finding the suitable hyperparameter relies on trial and test or knowledge from a specialist. In a scenario with numerous hyperparameters, this issue became a drawback. However, our clustering algorithm requires a single and intuitive hyperparameter, *m*.

The hyperparameter *m* is used by Autocloud to define the level of sensitivity applied to the samples received for processing. It will directly interfere with the algorithm, more or less stressing the process of detecting anomalies.

5 MATERIALS AND METHODS

This section presents the event logs, the encoding strategy, the clustering algorithm, and the metrics that we use to evaluate and validate our method's ability to detect anomalies in trace streams.

5.1 Event Logs

Essential subsidy to evaluate our method are the event logs, which contain the traces of business processes. First of all, we generated six different business process models: Gigantic, Huge, Large, Medium, Small, and Wide. For this process of designing business process models, we use the PLG2 tool [10]. This bunch of datasets was

generated to provide heterogeneous scenarios revealing possible drawbacks and advantages of trace clustering. In other words, we simulated different scenarios seeking particularities and barriers related to m hyperparameter.

In Table 1 it is possible to see details of the log files generated, specifying their sizes, percentage of anomalies, and business process models.

Model	#activities	#anomalies %	#traces	#trace len
Small	20	0,5,10,15 and 20	1,5 and 10k	10
Medium	32	0,5,10,15 and 20	1,5 and 10k	8
Large	42	0,5,10,15 and 20	1,5 and 10k	12
Huge	54	0,5,10,15 and 20	1,5 and 10k	11
Wide	34	0,5,10,15 and 20	1,5 and 10k	7
Gigantic	64	0,5,10,15 and 20	1,5 and 10k	11

Table 1: Overview of event log used in the experiments.

We started to generate the log files from the definition of the business process models to be used. Thus, we randomly generate the event logs from the models chosen and designed using the PLG2 tool. Following the control flow defined in the models, attributes were also generated for each event contained in the model [26].

In addition to the six types of business process models that supported the generation of event logs, we vary their respective sizes and number of anomalies. Also, it was generated a group of logs containing the same characteristics compromised with all the anomalies at once. The following is a description of six types of anomalies:

- Skip: Up to 3 activities are skipped
- Insert: Up to 3 random activities are inserted
- Rework: Up to 3 events have been run once more
- Early: A sequence of up to 2 events has been executed too early, and hence are skipped later in the case
- Late: 2 events were run later than usual
- Attribute: Up to 3 events receive incorrect attributes

5.2 Metrics

To evaluate the proposed approach, we tested a set of datasets with the clustering algorithm Autocloud. The metrics used in this comparison were Precision (1), Recall (2), and F1-Score (3).

$$Precision = \frac{TP}{TP + FN} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$F1 - Score = \frac{2.\ precision\ .\ recall}{precision + recall} \tag{3}$$

6 RESULTS AND DISCUSSION

We evaluated the application of clustering algorithms for detecting anomalies in business processes in trace streams, measuring their performance during this process.

Regarding the best value for the hyperparameter m, we suggest 2.5 as the most suitable value, as seen in Figure 2. Different from the suggested default value ($m = 3$), using 2.5 it was possible to achieve high-predictive detection, about 0.93% with a low F1-Score standard deviation (0.04). Values of m about 1, 1.5, and 2.5 lead to high variations in prediction performance. It is important to note that other PM scenarios could take advantage of a different m value, but we strongly suggest this value considering the massive number of experiments.

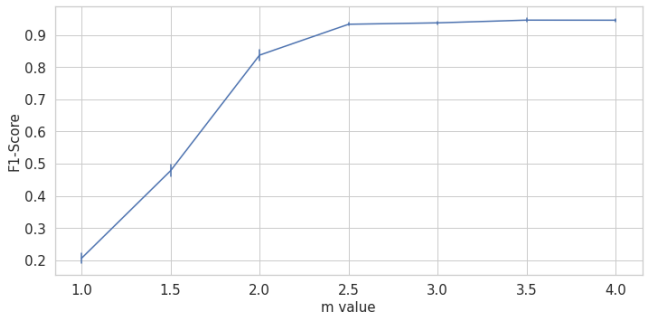

Figure 2: F1-Score across several values for hyperparameter m.

The following experiments took into account only m equal to 2.5. We explored five different incidences of anomalies (*None*, 5%, 10%, 15%, and 20%) throughout each evaluated stream. As Figure 3 shows, a stream without anomalies presented the best performance (1.0), as the incidence increases the performance in terms of F1-Score decreases, obtaining 1, 0.97, 0.95, 0.92 and 0.89 for *None*, 5%, 10%, 15% and 20%, respectively, as shown in Figure 3. We can see that the algorithm's performance in identifying anomalies is affected by the percentage of anomalies in the event logs.

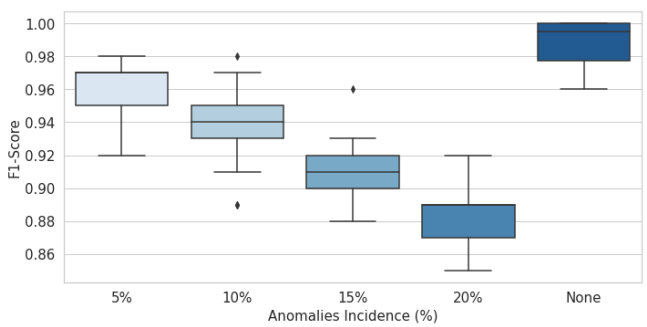

Figure 3: F1-Score obtained with Autocloud using $m = 2.5$ for anomaly detection with several percentage of anomalies (5%,10%,15%,20% and *none*) focusing on anomalies incidence.

The overall solution demonstrated consistent performance of about 0.97 for events logs containing 5% of anomalies of their total trace size, in terms of F1-Score for processing event logs with 1,000, 5,000, and 10,000 traces across five anomalies incidences. There is no change in performance considering different stream sizes. Autocloud obtained a very satisfactory performance in detecting anomalies and not identifying false anomalies, that is, false positives contained in the log files.

Regarding the traces, we explored all anomalies incidences with each particular trace file stream. The figure 4 shows the F1-Score achieved by all evaluated anomalies. Insert anomaly presented the highest detection rate, about 0.96 of F1-Score. This type of anomaly is one of the most traditional, also, can be easily detected by simple techniques, because its composition is not the most complex compared to the other types of anomalies used in the construction of our data sets. The other anomalies were also possible to be detected with a similarly high detection rate, in the following order: Insert (0.96), Early (0.95), Rework (0.95), Late (0.95), Skip (0.95), and Attribute (0.94). When we have all anomalies occurring in the same flow, our proposal obtained an F1-Score of 0.91.

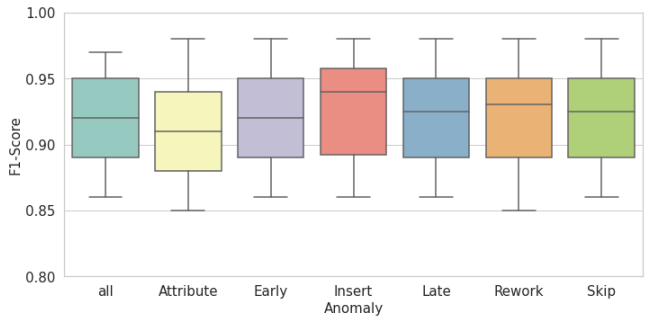

Figure 4: F1-Score obtained using Autocloud ($m = 2.5$) as anomaly detection system for seven different anomalies (all, Attribute, Early, Insert, Late, Rework and Skip) with different incidence levels.

Several characteristics affect the performance of anomaly detection. One of them is the process model. In our work, we explored six different processes (Gigantic, Huge, Large, Medium, Small, and Wide) to investigate the bias of a particular process when using anomaly detection systems. Figure 5 exposes some differences bringing important insights. A higher rate of detection was obtained from the Small process, obtaining an F1-Score about 0.95. The others obtained similar results: Gigantic (0.94), Huge (0.94), Large (0.92), Medium (0.92) and Wide (0.93).

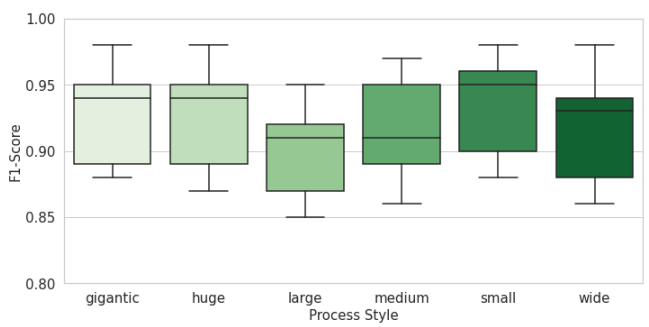

Figure 5: F1-Score obtained with Autocloud using $m = 2.5$ for anomaly detection with a perspective of process model and several process model (Gigantic, Huge, Large, Medium, Small and Wide).

Spearman coefficients for the F1-Score, anomaly type, anomaly incidence, and process model can be seen in Figure 6. A correlation equal to 1 indicates that there is a perfect relationship among two variables (-1 if a perfect reverse relation is observed). When the coefficients are close to 0, there is no evidence of correlation among the observed features. Comparing the three main features and F1-Score with itself will always generate a correlation coefficient equal to 1. In Figure 6, we can observe in the row F1-Score an average negative correlation with anomaly incidence, a low positive correlation with the process model, and a very low positive correlation with anomaly itself. Summing up, it is possible to mention that the incidence of anomaly is the most tricky phenomena to be dealt with when detecting anomalies on trace streams.

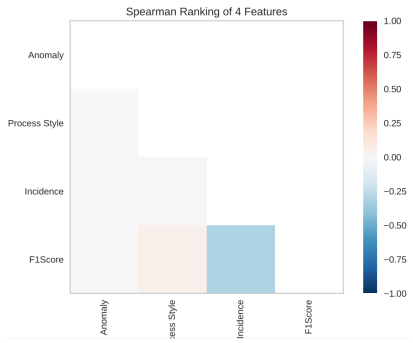

Figure 6: Spearman correlation coefficients among F1-Score, anomaly type, anomaly incidence and process model.

Another characteristic that affected, that is, compromises the performance of Autocloud during the execution of our experiments with this clustering algorithm was that the lower the value of m. We employed hyperparameter configuration ranging between 1 and 4), the lower the hyperparameter, the higher the detection sensitivity defined by this hyperparameter, consequently the lower the F1-Score. We can verify this behavior through graphs Figure 7 and Figure 8.

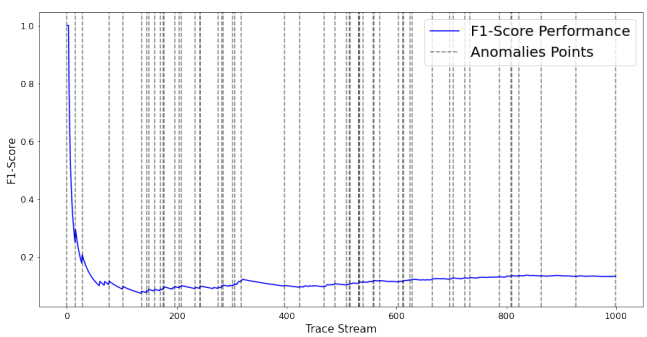

Figure 7: F1-Score with m=1.0 - 1k Cases - 5% anomaly

In the figures 7 and 8 the vertical dotted lines represent the lines that have some kind of anomaly, while the horizontal blue line represents the performance of the algorithm using the F1-Score metrics

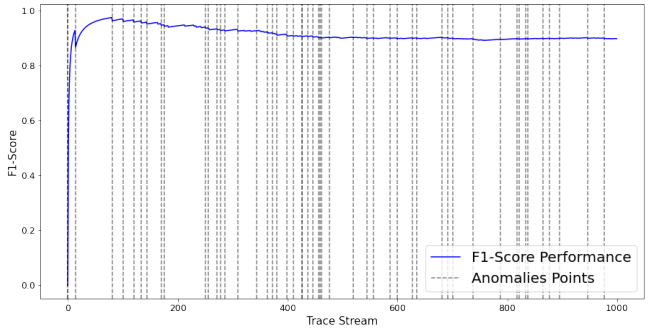

Figure 8: F1-Score with m=2.5 - 1k Cases - 5% anomaly

described in section 5. These figures show the behavior of the algorithm using the same data set ($10000 - TenPercentAnomalie - small - Insert - 10000 - 0.1 - 1.csv$), with the same incidence of anomalies. Only the value of the hyperparameter has been changed. Figure 7 with the value of $m = 1$ and figure 8 with the value of $m = 2.5$. Both results were obtained during the processing of the dataset: $10000 - TenPercentAnomalie - small - Insert - 10000 - 0.1 - 1.csv$, with the configuration of m in 1 and 2.5.

Based on the F1-Score numbers presented in the two graphs, we can see that the lower the value defined in the hyperparameter, that is, the higher the level of refinement of the algorithm, its performance falls when comparing the numbers obtained by the F1-Score concerning higher values in the hyperparameter m.

7 CONCLUSION

In this work, we proposed an online method for anomaly detection of process mining. We acquired traces from an event stream and extract a feature vector using the word2vec encoding. Clustering the feature vectors with the Autocloud algorithm, we were able to identify anomalous traces in an online way. We simulated and explored scenarios resembling reality for the experiments by constructing synthetic business process logs with different trace sizes, case numbers, and anomalies, a total of 630 event logs.

The capacity of Autoclud, identified through our experiments, paved the way for the automation of compliance and conformance checking for any type of company using the event log as a resource. The best performance of Autocloud was obtained with $m = 2.5$, the unique algorithm hyperparameter when detecting anomalies in the various dataset scenarios. This hyperparameter relies on the sensitivity of the clustering algorithm, very intuitively tuned. It is necessary to mention the other online clustering algorithms demand for several hyperparameters, so that, Autocloud was selected.

A similar F1-Score was observed regardless of the type of anomaly (Attribute, Early, Insert, Late, Rework and Skip) or event log complexity (Gigantic, Huge, Large, Medium, Small, and Wide). The most correlated phenomena related to the predictive performance was anomaly incidence since the high incidence (i.e., about 20%) at particular moments was interpreted as a typical behavior by a given data cloud.

As future work, we plan to compare the other clustering algorithms using a stream of events, reducing our framework's acquisition step. However, it will be necessary to use statistical assumptions to handle the incomplete event traces.

ACKNOWLEDGMENTS

The authors would like to thank CNPq (National Council for the Scientific and Technological Development) for their financial support under Grant of Project 420562/2018-4 and 309863/2020-1 and the program "Piano di sostegno alla ricerca2020" funded by Universit'a degli Studi di Milano

REFERENCES

[1] Wil van der Aalst. 2016. *Process Mining: Data Science in Action* (2 ed.). Springer-Verlag, Berlin Heidelberg. https://doi.org/10.1007/978-3-662-49851-4

[2] Marcel R. Ackermann, Marcus Märtens, Christoph Raupach, Kamil Swierkot, Christiane Lammersen, and Christian Sohler. 2012. StreamKM++: A clustering algorithm for data streams. *ACM Journal of Experimental Algorithmics* 17 (May 2012), 2.4:2.1–2.4:2.30. https://doi.org/10.1145/2133803.2184450

[3] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. 2003. A framework for clustering evolving data streams. In *Proceedings - 29th International Conference on Very Large Data Bases, VLDB 2003*. Morgan Kaufmann, 81–92. https://experts.illinois.edu/en/publications/a-framework-for-clustering-evolving-data-streams

[4] P. Angelov. 2014. Anomaly detection based on eccentricity analysis. In *2014 IEEE Symposium on Evolving and Autonomous Learning Systems (EALS)*. 1–8. https://doi.org/10.1109/EALS.2014.7009497

[5] Sylvio Barbon Junior, Gabriel Marques Tavares, Victor G Turrisi da Costa, Paolo Ceravolo, and Ernesto Damiani. 2018. A framework for human-in-the-loop monitoring of concept-drift detection in event log stream. In *Companion Proceedings of the The Web Conference 2018*. 319–326.

[6] Clauber Gomes Bezerra, Bruno Sielly Jales Costa, Luiz Affonso Guedes, and Plamen Parvanov Angelov. 2020. An evolving approach to data streams clustering based on typicality and eccentricity data analytics. *Information Sciences* 518 (May 2020), 13–28. https://doi.org/10.1016/j.ins.2019.12.022

[7] Fábio Bezerra and Jacques Wainer. 2013. Algorithms for anomaly detection of traces in logs of process aware information systems. *Information Systems* 38, 1 (March 2013), 33–44. https://doi.org/10.1016/j.is.2012.04.004

[8] F. Bezerra, J. Wainer, and Van Der W. M.P. Aalst. 2009. Anomaly detection using process mining. *Enterprise, Business-Process and Information Systems Modeling (10th International Workshop, BPMDS 2009, and 14th International Conference, EMMSAD 2009, held at CAiSE 2009, Amsterdam, The Netherlands, June 8-9, 2009. Proceedings)* (2009), 149–161. https://doi.org/10.1007/978-3-642-01862-6_13

[9] Kristof Böhmer and Stefanie Rinderle-Ma. 2016. Multi-perspective Anomaly Detection in Business Process Execution Events. In *On the Move to Meaningful Internet Systems: OTM 2016 Conferences*. Springer International Publishing, Cham, 80–98.

[10] Andrea Burattin. 2015. PLG2: Multiperspective Processes Randomization and Simulation for Online and Offline Settings. *ArXiv* (2015).

[11] Andrea Burattin and Josep Carmona. 2017. A Framework for Online Conformance Checking. In *International Conference on Business Process Management*. Springer, 165–177.

[12] Andrea Burattin, Sebastiaan J. van Zelst, Abel Armas-Cervantes, Boudewijn F. van Dongen, and Josep Carmona. 2018. Online Conformance Checking Using Behavioural Patterns. In *Business Process Management*, Mathias Weske, Marco Montali, Ingo Weber, and Jan vom Brocke (Eds.). Springer International Publishing, Cham, 250–267.

[13] Kristof Böhmer and Stefanie Rinderle-Ma. 2016. Multi-perspective Anomaly Detection in Business Process Execution Events. In *On the Move to Meaningful Internet Systems: OTM 2016 Conferences (Lecture Notes in Computer Science)*, Christophe Debruyne, Hervé Panetto, Robert Meersman, Tharam Dillon, eva Kühn, Declan O'Sullivan, and Claudio Agostino Ardagna (Eds.). Springer International Publishing, Cham, 80–98. https://doi.org/10.1007/978-3-319-48472-3_5

[14] F. Cao, M. Ester, W. Qian, and A. Zhou. 2006. Density-Based Clustering over an Evolving Data Stream with Noise. In *SDM*. https://doi.org/10.1137/1.9781611972764.29

[15] Josep Carmona, Boudewijn F. van Dongen, Andreas Solti, and Matthias Weidlich. 2018. *Conformance Checking - Relating Processes and Models*. Springer. 1–263 pages.

[16] Paolo Ceravolo, Gabriel Marques Tavares, Sylvio Barbon Junior, and Ernesto Damiani. 2020. Evaluation goals for online process mining: a concept drift perspective. *IEEE Transactions on Services Computing* (2020).

[17] Pieter De Koninck, Seppe vanden Broucke, and Jochen De Weerdt. 2018. act2vec, trace2vec, log2vec, and model2vec: Representation Learning for Business Processes. In *Business Process Management*, Mathias Weske, Marco Montali, Ingo Weber, and Jan vom Brocke (Eds.). Springer International Publishing, Cham, 305–321.

[18] Cleiton dos Santos Garcia, Alex Meincheim, Elio Ribeiro Faria Junior, Marcelo Rosano Dallagassa, Denise Maria Vecino Sato, Deborah Ribeiro Carvalho, Eduardo Alves Portela Santos, and Edson Emilio Scalabrin. 2019. Process mining techniques and applications – A systematic mapping study. *Expert Systems with Applications* 133 (2019), 260 – 295. https://doi.org/10.1016/j.eswa.2019.05.003

[19] Mohammadreza Fani Sani, Sebastiaan J. van Zelst, and Wil M. P. van der Aalst. 2020. Conformance Checking Approximation Using Subset Selection and Edit Distance. In *Advanced Information Systems Engineering*. Springer International Publishing, Cham, 234–251.

[20] Mohammed Ghesmoune, Mustapha Lebbah, and Hanene Azzag. 2016. State-of-the-art on clustering data streams. *Big Data Analytics* 1, 1 (Dec. 2016), 13. https://doi.org/10.1186/s41044-016-0011-3

[21] R. P. Jagadeesh Chandra Bose and Wil van der Aalst. 2010. Trace Alignment in Process Mining: Opportunities for Process Diagnostics. In *Business Process Management*. Springer Berlin Heidelberg, Berlin, Heidelberg, 227–242.

[22] S. B. Junior, P. Ceravolo, E. Damiani, N. J. Omori, and G. M. Tavares. 2020. Anomaly Detection on Event Logs with a Scarcity of Labels. In *2020 2nd International Conference on Process Mining (ICPM)*. 161–168. https://doi.org/10.1109/ICPM49681.2020.00032

[23] K. L. McMillan and D. K. Probst. 1995. A technique of state space search based on unfolding. *Formal Methods in System Design* 6, 1 (01 1 1995), 45–65. https://doi.org/10.1007/BF01384314

[24] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781 [cs]* (Sept. 2013). http://arxiv.org/abs/1301.3781 arXiv: 1301.3781.

[25] Maryam Mousavi, A. A. Bakar, and M. Vakilian. 2015. Data stream clustering algorithms: A review. https://www.semanticscholar.org/paper/Data-stream-clustering-algorithms%3A-A-review-Mousavi-Bakar/b78be4af0b68a47c0b3df3066d6ef95642f20552

[26] Timo Nolle, Stefan Luettgen, Alexander Seeliger, and Max Mühlhäuser. 2019. BINet: Multi-perspective business process anomaly classification. *Information Systems* (Oct. 2019), 101458. https://doi.org/10.1016/j.is.2019.101458

[27] A. Rozinat and W.M.P. van der Aalst. 2008. Conformance checking of processes based on monitoring real behavior. *Information Systems* 33, 1 (2008), 64 – 95.

[28] Daniel Schuster and Sebastiaan J. van Zelst. 2020. Online Process Monitoring Using Incremental State-Space Expansion: An Exact Algorithm. In *Business Process Management*, Dirk Fahland, Chiara Ghidini, Jörg Becker, and Marlon Dumas (Eds.). Springer International Publishing, Cham, 147–164.

[29] Gabriel Marques Tavares and Sylvio Barbon. 2020. Analysis of Language Inspired Trace Representation for Anomaly Detection. In *ADBIS, TPDL and EDA 2020 Common Workshops and Doctoral Consortium*, Ladjel Bellatreche, Mária Bieliková, Omar Boussaïd, Barbara Catania, Jérôme Darmont, Elena Demidova, Fabien Duchateau, Mark Hall, Tanja Merčun, Boris Novikov, Christos Papatheodorou, Thomas Risse, Oscar Romero, Lucile Sautot, Guilaine Talens, Robert Wrembel, and Maja Žumer (Eds.). Springer International Publishing, Cham, 296–308.

[30] G. M. Tavares, P. Ceravolo, V. G. Turrisi Da Costa, E. Damiani, and S. Barbon Junior. 2019. Overlapping Analytic Stages in Online Process Mining. In *2019 IEEE International Conference on Services Computing (SCC)*. 167–175. https://doi.org/10.1109/SCC.2019.00037 ISSN: 2474-2473.

[31] Gabriel Marques Tavares, Victor G Turrisi da Costa, Vinicius Eiji Martins, Paolo Ceravolo, and Sylvio Barbon Jr. 2018. Anomaly detection in business process based on data stream mining. In *Proceedings of the XIV Brazilian Symposium on Information Systems*. 1–8.

[32] Wil M. P. van der Aalst. 2016. *Process Mining: Data Science in Action* (2 ed.). Springer, Heidelberg.

[33] Sebastiaan J. van Zelst, Alfredo Bolt, Marwan Hassani, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. 2017. Online conformance checking: relating event streams to process models using prefix-alignments. *International Journal of Data Science and Analytics* (27 10 2017). https://doi.org/10.1007/s41060-017-0078-6

[34] Sebastiaan J. van Zelst, Mohammadreza Fani Sani, Alireza Ostovar, Raffaele Conforti, and Marcello La Rosa. 2020. Detection and removal of infrequent behavior from event streams of business processes. *Information Systems* 90 (2020), 101451. https://doi.org/10.1016/j.is.2019.101451 Advances in Information Systems Engineering Best Papers of CAiSE 2018.