A Framework for Human-in-the-loop Monitoring of Concept-drift Detection in Event Log Stream

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

Gabriel Marques Tavares
Londrina State University (UEL)
Londrina, Brazil
gtavares@uel.br

Victor G. Turrisi da Costa
Londrina State University (UEL)
Londrina, Brazil
victorturrisi@uel.br

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

Ernesto Damiani
Khalifa University (KUST)
Abu Dhabi, UAE
ernesto.damiani@kustar.ac.ae

ABSTRACT

One of the main challenges of Cognitive Computing (CC) is reacting to evolving environments in near-real time. Therefore, it is expected that CC models provide solutions by examining a summary of past history, rather than using full historical data. This strategy has significant benefits in terms of response time and space complexity but poses new challenges in term of concept-drift detection, where both long term and short terms dynamics should be taken into account. In this paper, we introduce the Concept-Drift in Event Stream Framework (CDESF) that addresses some of these challenges for data streams recording the execution of a Web-based business process. Thanks to CDESF support for feature transformation, we perform density clustering in the transformed feature space of the process event stream, observe track concept-drift over time and identify anomalous cases in the form of outliers. We validate our approach using logs of an e-healthcare process.

KEYWORDS

Process Mining, DBScan, Concept-drift, Clustering, Stream Mining

ACM Reference Format:
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 *WWW'18 Companion: The 2018 Web Conference, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3184558.3186343

1 INTRODUCTION

Recent advances in Cognitive Computing (CC) envision systems that can reason with purpose and interact with the environment, going a step forward the quantitative and deterministic approach of traditional Machine Learning [16]. A major challenge of CC is delivering autonomous reasoning and continuous learning toward rational comprehension and the natural interaction between humans and machines [24]. Tackling this challenge requires handling uncertain knowledge and approximate solutions [30], as well as being able to react to novel stimuli, avoiding abrupt degradation of performance [17]. If this reaction has to be organized in real time, or near-real time, we enter in the domain of data stream processing, where results have to be continuously updated using incoming updates.

A well-known problem affecting data streams is *concept-drift* where the underlying relations between a recorded tuple $\vec{x}$ and a system response y change over time [14, 18]. Ignoring concept-drift can lead to a deterioration in the quality of the algorithm and its capacity to represent the most recent concepts in data. Moreover, concept-drift is relevant to the validation process of any learning task because it recasts the ratio between observed data and response. Implementing a concept-drift adaptation strategy is not a trivial task as different types of concept-drift are possible and different adaptations can be operated in response to them [15].

In this paper, we argue that concept-drift may represent a key aspect to reinforce the cooperation between human and machines. Instead of creating an additional level of complexity that isolates the final user from the deep behavior of the system, concept-drift may offer a valid instrument to supervise its evolution and guide its validation via a human-in-the-loop process where the human observes the emergence of concept-drift over time and can address the system behavior by adjusting hyperparameters.

In order to achieve this goal, a concept-drift detection procedure has to implement the following properties: (a) be generically applicable, i.e. taking as input the results of any algorithm that process data streams; (b) be monitorable, i.e. a human can monitor it using a synthetic human-readable representation and navigate through the evolution of the system over time.

Along these lines, we introduce the Concept-Drift in Event Stream Framework (CDESF) to address concept-drift monitoring for a generic data stream processing algorithm. To make the system evolution understandable to the human user, CDESF encodes it using a three dimensional space whose axes are time and two metrics summarising the results of the analytics applied to incoming data. When orthogonal aspects are captured by the two selected metrics, the user obtains a wide coverage of the response offered by the algorithm. At the same time, when the probability distribution of responses changes, the user observes an evolution in the CDESF representation. Besides displaying the evolution, our CDESF acts as a controller on hyperparameters handling concept-drift detection; in particular, it can modify the probability distribution adopted to identify a concept-drift.

To evaluate CDESF, we identified a concrete learning task and a reference literature to compare our results. We chose to address

WWW'18 Companion, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5640-4/18/04..
https://doi.org/10.1145/3184558.3186343

Process Mining (PM) [26] because it imposes challenging constrain to event stream processing [6] and because change detection is a widely addressed topic [2, 8].

The remainder of the paper is organised as follows. Section 2 presents the overall framework proposed in this paper. This section also discusses some formal concepts of process mining, stream analysis, anomaly detection and trace clustering. In this section, stream clustering challenges are presented and together with some human-friendly insights obtained from the proposed framework. After that, Section 3 describes the clustering algorithms used in the experiments. Section 4 presents the technologies involved in the implementation and evaluation of the framework. Finally, Section 5 concludes the paper and outlines our future work.

2 EVENT STREAM FRAMEWORK

This section describes the workflow of the proposed framework, as well as, the technologies and techniques applied. In Figure 1 it is possible to see an overview of CDESF.

2.1 Formalization

2.1.1 Process Mining Essentials. The learning task addressed in this work is PM. PM exploits event streams to run analytics on business processes. Unlike other stream-related problems, PM analysis cannot be approached with traditional stream mining techniques, since the single tuple imputing the learning procedure is obtained by grouping multiple events through time, with past events affecting future ones [5]. In PM each event records the execution of some activity, which is a task or an action carried out by the user. Each event is also related to a single instance or execution, often called *case* [27], which gathers the sequence of activities executed to achieve a specific result. A unique sequence of activities is called a *trace*; thus multiple cases can exhibit the same trace.

Events can then be ordered based on their time of execution or grouped based on their activity type, or other attributes, such as the cost of execution of an activity, the originator (who started or supervised an execution), or the resources exploited during execution.

2.1.2 Core Data Stream concepts. In traditional machine learning, one has access to all available data, however, when dealing with data streams, this assumption cannot be made. In this scenario, data is usually made available through infinite streams in which underlying regularities of data may evolve during time [14, 18]. A stream S is defined in the format $S = \{i_1, i_2, i_3, ..., i_n\}$, where i corresponds to a pair $(\vec{x}, y)$ when the ground truth for that instance is known or simply $\vec{x}$ when it is not, with $\vec{x}$ being the feature vector of that instance and y being its label, and n is possibly infinite.

Since data processing is continuous and potentially infinite, it is not feasible to store all observed instances for future computations. Metrics and statistics have then been used to cope with single-time processing [11, 14, 19]. Histograms, for example, can be computed in an online fashion, but the information they provide is no more than a summary of what the system observed.

CDESF addresses data stream processing using an informed forgetting mechanism. Older cases are deleted at time horizons adjusted by a Nyquist rate [10] that define the minimum number of instances required to properly update the system mode. The following subsection gives more details about our forgetting mechanism as a whole.

2.1.3 Hyperparameters. Since ingestion of event data is asynchronous, we cannot rely on our framework to start with an event data set. For this reason, inspired by the concept of Grace Period (GP) presented in [11], we introduced the idea of a period where data is collected to bootstrap it. It means that during the GP, there is no reference model and new events are used to feed the model construction. GP is a hyperparameter for our framework and different GP values impact in different modelling and following analysis. For example, if the GP is ten, the framework will render the stream until ten distinct cases are available. Then the GP is declared over, and the model creation is triggered. From this point on, each new event is processed, compared to the model and evaluated.

Time horizon (TH) is a hyperparameter that specifies a time interval in seconds. We refer to the end of a TH as a checkpoint (CP). Thus, at CP the framework will reevaluate its number of cases used to update the histograms. The exact number of cases is calculated by the Nyquist sampling theorem, where the sampling frequency during data acquisition should be at least twice the highest frequency contained in the signal [20]. Figure 2 shows an example of histogram updating using a time horizon of 60 seconds.

At each CP, the number of total cases is verified to check if there are no more cases than the maximum delimited by the Nyquist parameter. If there are more cases than Nyquist requires, the older cases are released from memory and a new Nyquist value is generated. The highest frequency in our method is the number of new cases that occurred during the last TH. If the new Nyquist is smaller than the initial GP number, then it is set to its initial value (which is the GP). A flag is raised because it would not be cohesive to create metrics with a lower number of cases than the GP. Thus, with a new set of cases, determined by the Nyquist theorem, metrics are updated. Equation 1 shows our adaptation of the Nyquist frequency.

$$Nyquist\ Frequency = number\_of\_new\_cases * 2 \quad (1)$$

2.1.4 Case analysis. After the end of GP, a model is constructed with the goal of evaluating traces and pointing out the irregular and common ones. The metrics we adopted to control the system behavior are based on a histogram of traces and a histogram of timestamps. The histogram of traces counts the number of occurrences of each event throughout all cases of a specific process, giving us information about the recurrence of activities.

When new events arrive, the corresponding case is retrieved, and its activity is added to the trace. The updated trace string is then compared to the histogram string. Since this is a string-to-string matching problem, we used the well-known Edit Distance algorithm [28], which aims to compare two strings and quantify their dissimilarity. The standard edit distance algorithm allows three edit computations: changing one symbol of a string into another single one; deleting one symbol from a string; or inserting a single symbol into a string. Our string comparison does not depend on the order of the characters, so only two of the three edit operations (deletion and insertion) are relevant.

After identifying events that are different in both strings (histogram and trace), a weighted value based on their histogram occurrence is calculated. The sum of the weighted distances is the final value of the trace comparison with the histogram.

We named this new distance calculation as Edit Weighted Distance (EWD). EWD is a key notion in our approach and is considered (along with time analysis) a decisive descriptor of a case's behavior. An example follows to make this idea clearer.

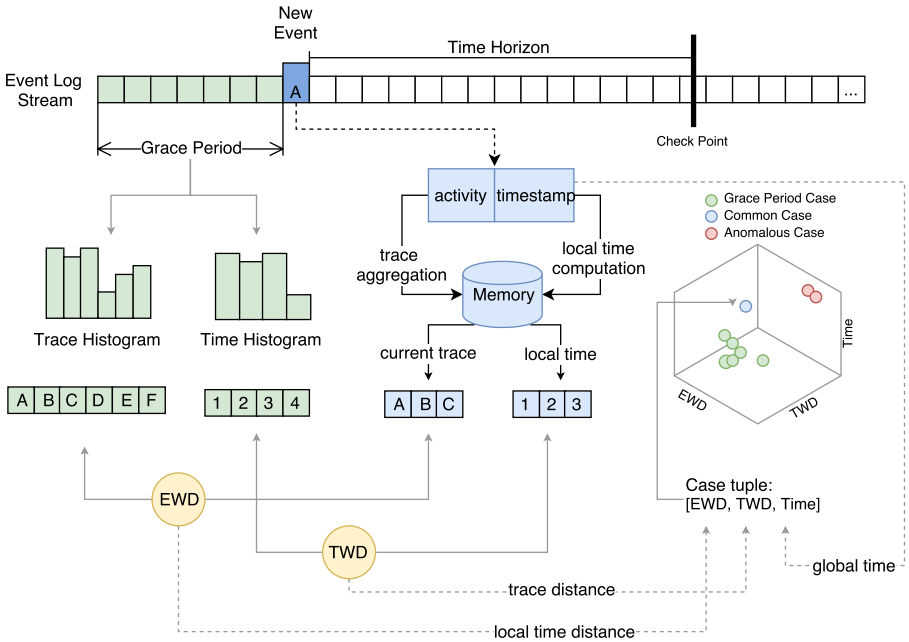

Figure 1: Overview of Stream Process Mining Framework

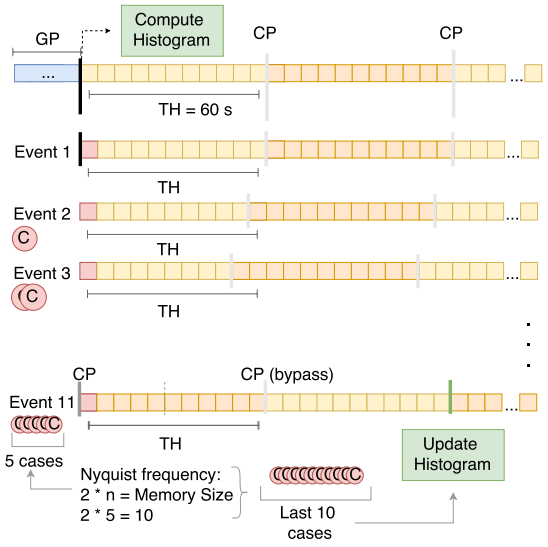

Figure 2: Histogram computation. After 11 events the histogram was updated, the memory adjusted to keep the last 10 cases and a checkpoint (CP).

Given a set of traces $L = \{\langle a, b, c, d, e\rangle, \langle a, b, c\rangle, \langle a, d, c\rangle, \langle a, b\rangle\}$, a histogram H is built from the events' frequencies in L. Thus, $H = \{4, 3, 3, 2, 1\}$. The order of the histogram values follows the alphabetical order of the letters. Given a new trace $T = \langle a, b, c\rangle$, the EWD value is given by the weighted distance between the two strings. The different symbols between H and T are $\langle d, e\rangle$. The weighted distance comes from the normalization (Equation 2) of H, which rescales the values into a range of $[0, 1]$; so, in this case, $Hnorm = \{1, 0.75, 0.75, 0.5, 0\}$. The EWD value is the sum of the correspondent d and e events, thus, EWD = 0.5.

$$Xnorm = \frac{X - Xmin}{Xmax - Xmin} \tag{2}$$

In the case of an activity that is present in T and not in H, its weighted value is determined as 0.5.

The histogram of timestamps is built from the same group of cases used for the histogram of traces but with some additional steps. First, for each case, a list of the differences between the events timestamps is created. Then, the list serves as input for quartiles calculation. Quartiles are cutpoints dividing the range of a probability distribution with equal probabilities [13]. Lastly, the list values are placed into the quartile bins. We explain our binning techniques with reference to Table 1, which represents several events from the same case.

- The events timestamps are arranged in a list, i.e. [2012/12/14 19:52:39, 2012/12/14 20:34:00, 2012/12/14 23:23:20, 2012/12/15 01:42:51, 2012/12/15 07:28:00, 2012/12/15 11:55:05];
- The time difference between activities is calculated in seconds, resulting in [0, 2481, 12641, 21012, 41721, 57746];
- The quartiles are computed based on the time distance: [0, 5021, 16826.5, 36543.75, 57746];
- The time differences are binned, i.e. put into the quartiles bins by range. Example: 0 falls between [0, 5021], so it is placed in the first quartile. The same applies to 2481. Then, 12641 is between [5021, 16826.5], so it belongs to the second quartile. This is done until the last value of the time differences;
- The resulting bin is [2, 1, 1, 2]. This means that the first quartile has two elements, the second has one and so on.

The same steps described above are repeated for all cases. Finally, the histogram is created from the sum of the bins.

Case ID	Activity	Complete Timestamp
Case 55	A	2012/12/14 19:52:39
Case 55	B	2012/12/14 20:34:00
Case 55	C	2012/12/14 23:23:20
Case 55	D	2012/12/15 01:42:51
Case 55	E	2012/12/15 07:28:00
Case 55	F	2012/12/15 11:55:05

Table 1: Log of events from the same case

Given a new event, its case timestamps are retrieved and binned. The bin is normalised and subtracted from the (also normalised) histogram of timestamps (both normalisations follow Equation 2). The result is the time-weighted distance (TWD) related to local time representation, which considers the interval between the activities of a case. On the other hand, the global time concerns to the last event of a given case. Thus, the local time is represented by a normalised distance from the histogram and the global time obey the real-event timestamp.

In general terms, the weighted distances - EWD (trace) and TWD (local time) - and global time are parameters that describe the behavior of a given case projected in the feature space. These three features compose a triple:

$$\langle EWD, TWD, Time \rangle$$

summarising a given case.

2.1.5 Concept-Drift and Anomaly Detection. Concept-Drift and Anomaly detection are two connected tasks. An anomalous phenomenon is related to the absence or presence of patterns in the event log which may indicate fraudulent actions, security violation or a malfunction in the system [2]. Frequently, a process anomaly related to a probing or attack, when detected early enough, can be halted or migrated to a honeypot [1]. Available techniques for anomaly detection include techniques related to density-based [4], nearest neighbours [29], and partition-based [23]. Outliers are detected in the case's feature space (EWD, TWD and time) by looking at distribution sparsity.

Clustering is a popular technique for detecting anomalies [25]. When a new sample falls outside the boundary of any existing cluster, it is marked as an anomaly and its density is monitored. We remark that an increase in the number of samples within the radius of an anomalous one indicates a concept-drift. Our technique can be considered an implicit drift detector since it relies on the unlabelled cases' feature values. Not needing labeled examples makes our technique useful in applications where labelling is expensive, time-consuming or not possible at all [25].

In Figure 3, five anomalous micro-clusters are highlighted in feature space based on trace distance, local time distance and global time. This snapshot was obtained after 3700 events and each point represents a case. The current event (the orange point) is `CHANGE DIAGN` from the web-based process *Hospital_Billing* which corresponds to someone using the application for modifying a patient's diagnosis. The micro-clusters correspond to unusual event sequences within the application log. In the next Section, we will discuss how to facilitate micro-cluster interpretation on the part of humans.

2.2 Human-in-the-loop monitoring

Analysing logs at the granularity of an individual event might not always allow the perceptions of phenomena like anomalies [2].

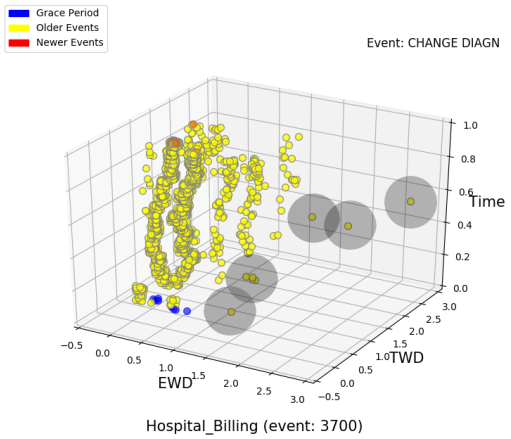

Figure 3: Five anomalous micro-clusters highlighted from feature space based on trace distance (EWD), local time distance (TWD) and global time after 3700 events.

Moreover, the lack of interoperability related to context information certainly occurs from a direct event log stream analysis. We face the event granularity drawback aggregating the events as a trace and extracting from it only three features (EWD, TWD and time). EWD is computed directly from the trace, as wide explored in PM solutions [7, 26]. In [3] the use of local (TWD) and global (time) features based on time were used in statistical hypothesis to discover concept-drifts and enrich the trace analysis. By these features, the CDESF can improve understanding and promote fact-based insights from the processes.

A human analyst is interested in knowing whether there are any unusual patterns in the log [2]. Specifically, clustering data relies on humans to become aware of a chance and explain its significance referring to topological structured groups of data. Even more, this would help when dealing with a massive amount of data where change is not easily noticeable due to the infrequent occurrence of anomalous events.

Our technique transforms events stream into cases and reducing their information to three features, providing a clustering strategy of anomaly detection. Furthermore, the clustering stream provides continuous interactions between human and computer during the data processing. As we shall see, when dealing with three dimensions alone, it is possible to construct a human-friendly inference system.

3 CLUSTERING ALGORITHMS

There are two main types of clustering algorithms: partitioning and hierarchical. In the first one, the algorithm tries to group instances while trying to minimise a goal function, for example, to create the most homogeneous k clusters, with k being a hyperparameter. On the other hand, hierarchical algorithms build a top-down, or bottom-up, approach by starting with a single cluster with all instances and performing splits until each instance has its own cluster. Or the other way around, merging clusters until a single cluster exists. The choice of when to stop splitting is also a hyperparameter very difficult to determine [12].

Partitioning algorithms, for example K-means, are generally incapable of dealing with clusters with arbitrary shapes [9]. To deal

with this, and to remove the need to choose a good k value according to domain knowledge, and outliers, we used the DBSCAN algorithm [12].

This algorithm works by starting with an arbitrary instance and expanding its cluster according to density-based metrics. Given two hyperparameters, n_min and ϵ, which corresponds to the minimum number of instances to build a cluster and the maximum density difference between two clusters when merging them, the algorithm expands regions until all instances are contained in a cluster or they are considered outliers and cannot be part of any group. Lastly, the average complexity of DBSCAN is $O(n\,log(n))$, which fits with stream processing requirements. When using DBSCAN, one needs to optimise the ϵ value, since it is the most sensitive hyperparameter of the two. Let the distance between two sets of points S1 and S2 be equal to $min\{dist(p,q) \mid p \in S1, q \in S2\}$. Then, the two sets having at least the density of the least dense cluster in the database will be separated from each other only if the distance between the two sets is larger than ϵ [12]. Since the density can be computed from unlabeled data only, it could be used as a substitute to explicitly labeled drift detection techniques, for monitoring changes [25].

4 IMPLEMENTATION AND EVALUATION

Our framework was implemented in Python (version 3). The source code and some demo event log are publicly available [1]. For the DBSCAN algorithm, we used the Scikit Learn library [22].

The choice of data set was guided by a real-life healthcare scenario with a massive stream [2]. The event log is formed by events (`NEW`, `CHANGE DIAGN`, `FIN`, `RELEASE`, `CODE OK`, `CODE NOK`, `STORNO`, `REJECT`, `REOPEN`, `DELETE`, and `BILLED`) that are related to the billing of medical services. Each trace of the event log records the activities executed to bill a package of medical services that were bundled together [21]. The event log is composed of 451.359 anonymised events (activities) which comprehend more than 100,000 traces random sampled of process instances that were recorded throughout three years. The time between events within a trace has not been altered.

We reported the results in the first month of hospital event log to make a concise exploration through comparing with [21] results. There is a large number of events and cases per day to support TH hyperparameter exploration. TH variation translates into a more or less recurrent CPs, which influences the histogram update. There are few days with less than 10 cases, with the minimum of cases per day being 5. This particular day had seven events, which poses as the minimum of events per day. The most active day in terms of events had 139; the mean stands at 72 with a standard variation of 33. The trace length is also important. Even though the most extended trace is about nine events, most of the traces are way smaller, with a mean of 1.6 and a standard deviation of almost 0.8. Time lengths, on the other hand, are usually considerable. The longest case goes on for almost 26 days, which is understandable when considering a hospital treatment. However, the mean time length is around 0.76 days with a standard variation of 3.12. There are several cases with only one event, so there is no time difference between the beginning and the end of the case.

In Manhard et al. [21] the same healthcare data set was explored using the Data-aware Heuristic Miner (DHM). DHM discovered a model (Figure 4) which fits 97% of the observed behavior in the the beginning of event log. This result emphasises two interesting aspects: the presence of anomalous processes and the challenging task to deal with the data set by a static approach. The authors made some assumptions about the behavior of specific activities supported by interviews with the hospital domain.

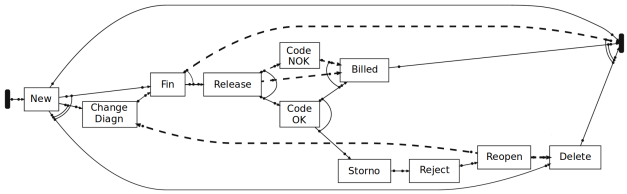

Figure 4: Process model of DHM [21] with 97% of fitting

In order to evaluate our framework's performance, different time horizons (6 h, 12 h, 24 h, 48 h, and 96 h) were explored with $n\_min = 1$ and $\epsilon = 0.1$ as hyperparameters of DBSCAN. It was possible to detect anomalies and concept-drifts in the first part of the stream as described in [21]. The great part of anomalies was detected from 114 h to 240 h of processing stream, as Figure 5 shows. The red line exposes a peak of anomalous cases in some TH values (12 h, 24 h, 48 h, 96 h). TH value of 6 h differs from the other TH values in detecting anomalous cases since the two greatest amount of outliers were detected before 110 h and after 618 h. This occurs due to a horizon that undersampled the events necessary to support assumptions about this stream scenario. In other words, for this scenario, a time horizon superior to 24 h fits well to anomaly detection. More about anomalous cases are explored in the Section 4.1.

In Figure 5 it is also possible to observe the concept-drift phenomenon. The black vertical line in the figure points the identification of a drift, which could be automatically discovered by observing the time the number of outliers detected is below the number of clusters generated. This occurs once the cluster is updated from the recomputed histogram, and an outlier case converges to a common cluster. This convergence concerns a change in the global density of feature space with a cluster region expansion towards grouping an outlier case. Concept-drift, as anomalies, were better detected with higher time horizons. This concept changing is exposed with more details in Section 4.2

4.1 Anomaly detection results

A close look at the first two outliers detected exemplifies the CDESF anomalous cases detection over the stream. In the figures 6, 7, 8, 9, 10 it is possible to see the different time horizons and the same anomalous cases: Case 29 and 291 (*Hospital_Billing*). In other words, the anomalies were recognised by ranging the TH hyperparameter from 6 h to 96 h and a suspicious behavior was discovered. In Table 2 it is possible to observe the details of trace and time histograms of CPs including the distances (EWD and TWD) from these anomalous cases. The cluster of common cases presented an average EWD and TWD of 0.39 and 0.55, respectively. As shown in Table 2, anomalous cases presented significantly superior values.

An important consideration about Case 29 is related to its recent activity when compared to Case 291. Case 29 is positioned in the top of the snapshots, independent of TH, due to its last related event being more recent to the Case 291. This can support the indication of suspicious time spent in Case 29. However, Case 291 presented a higher TWD than other detected anomalies, emphasising a more uncommon behavior. It is worth noting that Case 291 trace sequence

[1] https://github.com/gbrltv/CDESF
[2] https://data.4tu.nl/repository/uuid:76c46b83-c930-4798-a1c9-4be94dfeb741

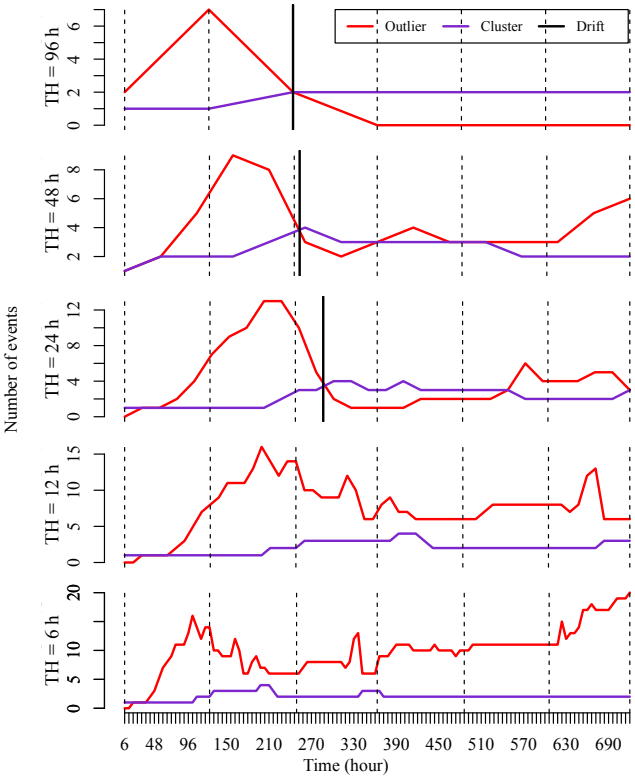

Figure 5: Flow analysis of thirty days of event log stream. Time Horizon (TH) of 24 h, 48 h and 96 h were capable to detect a concept-drift (highlighted by a black vertical line)

TH	CP	Trace Hist[1]	Case 29		Case 291	
			Trace	EWD	Trace[2]	EWD
6	13	[a:19,c:10,d:1,e:1]	[a, d, e]	2	[a, b, d, e]	1.5
12	7	[a:20,c:5]	[a, d, e]	2	[a, b, d, e]	1
24	4	[a:36,c:8,d:1,e:1]	[a, d, e]	1.2	[a, b, d, e]	1
48	2	[a:10,c:2]	[a, d, e]	2	[a, b, d, e]	1.3
96	1	[a:10,b:1,c:4]	[a, d, e]	1.3	[a, b, d, e]	1.4
		Time Hist	Time	TWD	Time	TWD
6	13	[20, 0, 0, 11]	[1, 0, 0, 1]	0.61	[1, 1, 0, 1]	1.4
12	7	[20, 0, 0, 5]	[1, 0, 0, 1]	0.8	[1, 1, 0, 1]	1.56
24	4	[36, 1, 0, 9]	[1, 0, 0, 1]	0.8	[1, 1, 0, 1]	1.56
48	2	[10, 0, 0, 2]	[1, 0, 0, 1]	0.8	[1, 1, 0, 1]	1.5
96	1	[10, 0, 0, 5]	[1, 1, 0, 1]	1.5	[1, 1, 0, 1]	1.5

[1] Conversion of activity names and letters: 'NEW': 'a', 'DELETE': 'b', 'CHANGE DIAGN': 'c', 'FIN': 'd', 'RELEASE': 'e', 'CODE OK': 'f'
[2] The trace sequence acquired was a, d, e, and b

Table 2: Case 29 and 291 identified as anomalous based on several time horizons

was a, d, e and b. However, our approach does not rely on the sequence, and the histogram would have been the same with other permutations of the same group of activities. Also, old cases are released from memory between CPs using the Nyquist rate.

The authors in [21] presented some important diagnosis about the cases. Deleted cases should not be in the closed status, whereas reopened cases with a change in diagnosis can be eventually closed in the future. Case 29 runs all activities (`NEW`, `FIN`, `RELEASE`, `CODE OK` and `BILLED`), but spends a long time to reach the last activity. Case 291 clearly represents an anomaly, since it did not follow the

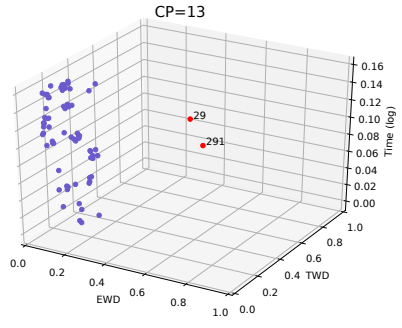

Figure 6: Snapshot obtained using 6 hours as TH, cases 29 and 291 detected as anomalies

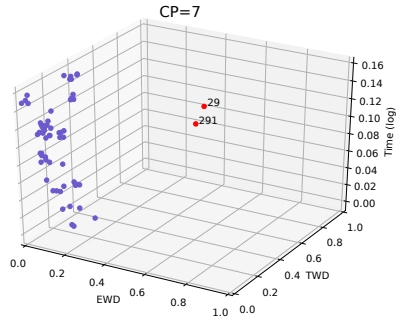

Figure 7: Snapshot obtained using 12 hours as TH, cases 29 and 291 detected as anomalies

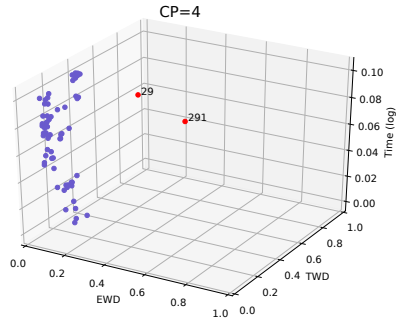

Figure 8: Snapshot obtained using 24 hours as TH, cases 29 and 291 detected as anomalies

common *Hospital_Billing* process, running `NEW`, `FIN`, `RELEASE` and `DELETE`.

4.2 Concept-drift issues

It was possible to detect the same concept-drift by three different time horizons. As possible to observe in Figure 5, the checkpoints 3 (96 h), 6 (48 h) and 13 (24 h) exposed a CD phenomenon in the stream. This CD was capable of changing several outliers with TH = 24 h (cases 71, 294 and 308) and with TH = 48 h (cases 29, 71, 101, 155 and 525) to be faced as common cases. The Case 71 is an example, it was identified as an outlier until histogram updating (CP=12 for TH=24 and CP=5 for TH=48) and after clusterized as a common case, as Figures 11 and 12 show.

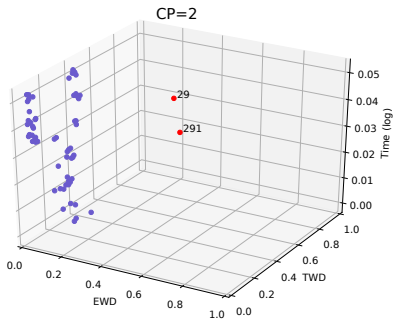

Figure 9: Snapshot obtained using 48 hours as TH, cases 29 and 291 detected as anomalies

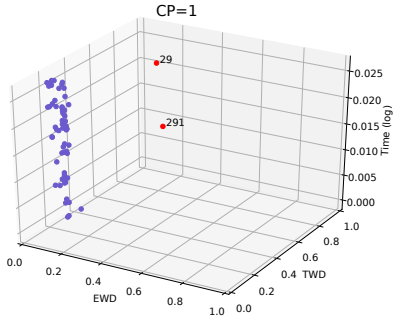

Figure 10: Snapshot obtained using 96 hours as TH, cases 29 and 291 detected as anomalies

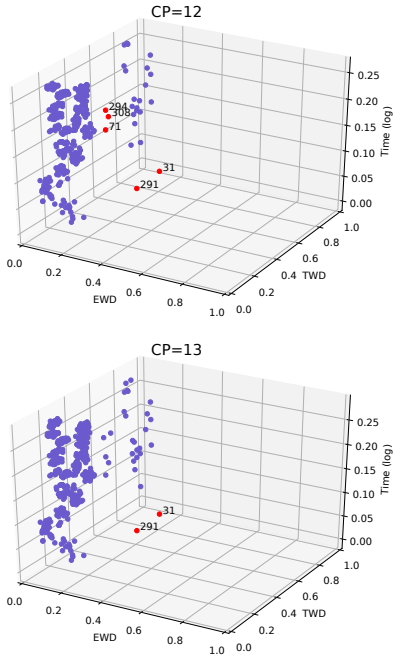

Figure 11: Concept-drift with time horizon of 24 hours

This fact occurs due to the adaption of the histogram by new cases similar to the former outlier towards discovering its pattern as a common one. The time horizon influences directly the anomaly detection since a rather wide horizon carries on several anomalous cases ending with aggregating them as a common cluster. On the other hand, a narrow limit increases the number of false positive cases.

As Figure 11 shows, Case 71 changed from an anomalous to common behavior between CP 12 and 13 with a TH = 24 h. Table 3 exposes this change. The EWD value went from 1.04 to 0.04. This is a clear picture of a concept-drift. That is, the case's behavior was far from the other cases, however more cases that mimic Case 71 nature appeared between those CPs, and that influenced the histogram update, which made Case 71 come to the common group. The same phenomenon happened with a TH of 48 hours (Figure 12), but both EWD and TWD values were maintained. This is explained by the fact that a group of cases with similar nature appeared between the CPs, but they were not significant enough to change the histogram density completely. However, the group as a whole characterizes new behavior in the stream, which put them as common cases. The contrasting example is Case 291, which presented as an anomaly that kept is pattern after a concept-drift in several experimented time horizons.

TH	CP	Trace Hist[1]	Case 71		Case 291	
			Trace	EWD	Trace[2]	EWD
24	12	[a:81,b:3,c:39,d:5,e:4,f:1]	[a, c]	1.04	[a, b, d, e]	1.9
24	13	[a:94,b:3,c:53,d:1,e:1,f:2]	[a, c]	0.04	[a, b, d, e]	0.8
48	5	[a:106,b:1,c:48,d:5,e:3]	[a, c]	1.05	[a, b, d, e]	1.5
48	6	[a:133,b:4,c:63,d:7,e:5,f:1]	[a, c]	1.05	[a, b, d, e]	0.6
		Time Hist	Time	TWD	Time	TWD
24	12	[85, 6, 1, 41]	[1, 0, 0, 0]	0.5	[1, 0, 0, 0]	0.4
24	13	[99, 4, 0, 51]	[1, 0, 0, 0]	0.5	[1, 0, 0, 0]	1.4
48	5	[108, 4, 0, 51]	[1, 0, 0, 0]	0.5	[1, 0, 0, 0]	0.2
48	6	[137, 7, 1, 68]	[1, 0, 0, 0]	0.5	[1, 0, 0, 0]	1.4

[1] Conversion of activity names and letters: 'NEW': 'a', 'DELETE': 'b', 'CHANGE DIAGN': 'c', 'FIN': 'd', 'RELEASE': 'e', 'CODE OK': 'f'
[2] The trace sequence acquired was a, d, e and b

Table 3: Concept-drift and cases 71 and 291.

TH selection is dependent on the problem and stream evaluated. In the case of the hospital stream, best results were obtained by a TH with at least 24 hours. As Figure 5 shows, time horizon of 6 and 12 hours are not able to detect concept-drifts, this is due to the lower CP time, and the consequence is a biased adaptation. Most business processes require a quite large time of observation. Thus, with a larger window (horizon), CDDSF can better understand the standard behavior, detect anomalies and fit itself over time.

It is important to highlight that CDDSF could deal with incomplete traces through the proposed feature space, where the traces that did not reach the final common activity are grouped in the same cluster. These aspects support our purpose of a reacting system, where a case on the fly (even incomplete) could be distinguished. Furthermore, a running process case could be followed closely when presenting an anomalous pattern since the first activities.

5 CONCLUSION

In this paper, we have addressed the problem of reacting to evolving environments in near-real time. For this purpose, we have developed a new framework to identify anomalies and concept-drift through a given time horizon desired by the specialist. In order to validate our approach, we have carried out various experiments using a real-life healthcare stream data set.

We have noticed the ability of our framework to effectively detect the anomalies and drift detection over the stream with different user time horizons. Even more, an unfinished process case could be

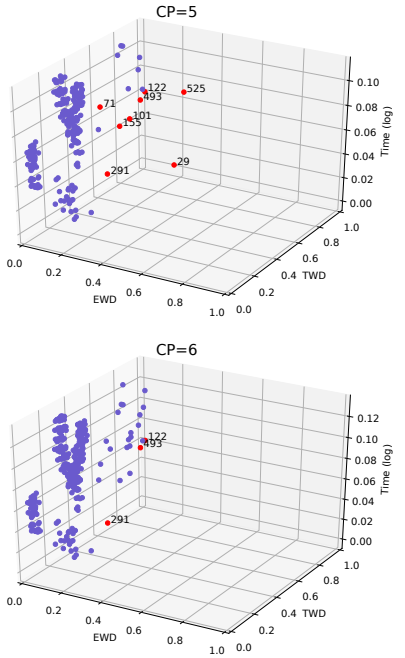

Figure 12: Concept-drift with time horizon of 48 hours

observed closely from the beginning leading to early identification of anomalous patterns. This way, it is possible to mitigate a costly error, resist an attack before it reaches its goal, halt or migrate the fraudulent execution to a honeypot.

For future work, we aim at developing our framework toward handling complex concept-drift by monitoring unsupervised indicators, in order to detect change earlier. This can be of interest for many real-world applications where some drift phenomena need to be prematurely identified.

ACKNOWLEDGMENTS

The authors would like to thank the Information and Communication Technology (ICT) Fund. ABU DHABI for the financial support for this research.

REFERENCES

[1] Kristof Böhmer and Stefanie Rinderle-Ma. 2016. Automatic signature generation for anomaly detection in business process instance data. In *International Workshop on Business Process Modeling, Development and Support*. Springer, 196–211.

[2] RP Jagadeesh Chandra Bose and Wil MP van der Aalst. 2010. Trace Alignment in Process Mining: Opportunities for Process Diagnostics.. In *BPM*, Vol. 6336. Springer, 227–242.

[3] RP Jagadeesh Chandra Bose, Wil MP van der Aalst, Indrė Žliobaitė, and Mykola Pechenizkiy. 2011. Handling concept drift in process mining. In *International Conference on Advanced Information Systems Engineering*. Springer, 391–405.

[4] Markus Breunig, Hans-Peter Kriegel, Raymond Ng, and Jörg Sander. 1999. Optics-of: Identifying local outliers. *Principles of data mining and knowledge discovery* (1999), 262–270.

[5] Andrea Burattin, Marta Cimitile, Fabrizio M Maggi, and Alessandro Sperduti. 2015. Online discovery of declarative process models from event streams. *IEEE Transactions on Services Computing* 8, 6 (2015), 833–846.

[6] Andrea Burattin, Alessandro Sperduti, and Wil MP van der Aalst. 2014. Control-flow discovery from event streams. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*. IEEE, 2420–2427.

[7] Paolo Ceravolo, Ernesto Damiani, Mohammadsadegh Torabi, and Sylvio Barbon. 2017. Toward a New Generation of Log Pre-processing Methods for Process Mining. In *International Conference on Business Process Management*. Springer, 55–70.

[8] Paolo Ceravolo, Ernesto Damiani, and Marco Viviani. 2005. Adding a peer-to-peer trust layer to metadata generators. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer, 809–815.

[9] Yixin Chen and Li Tu. 2007. Density-based clustering for real-time stream data. *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining KDD 07* d (2007), 133. https://doi.org/10.1145/1281192.1281210

[10] RA DeCarlo, J Murray, and R Saeks. 1977. Multivariable nyquist theory. *Internat. J. Control* 25, 5 (1977), 657–675.

[11] Pedro Domingos and Geoff Hulten. 2000. Mining high-speed data streams. *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining* (2000), 71–80. https://doi.org/10.1145/347090.347107

[12] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A Density-based Algorithm for Discovering Clusters a Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96)*. AAAI Press, 226–231.

[13] Michael Frigge, David C Hoaglin, and Boris Iglewicz. 1989. Some implementations of the boxplot. *The American Statistician* 43, 1 (1989), 50–54.

[14] João Gama, Pedro Pereira Rodrigues, Eduardo Spinosa, and Andre Carvalho. 2010. Knowledge Discovery from Data Streams. *Web Intelligence and Security - Advances in Data and Text Mining Techniques for Detecting and Preventing Terrorist Activities on the Web* (2010), 125–138. https://doi.org/10.3233/978-1-60750-611-9-125

[15] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)* 46, 4 (2014), 44.

[16] Xiaoxi Huang, Huaxin Huang, Beishui Liao, and Cihua Xu. 2013. An ontology-based approach to metaphor cognitive computation. *Minds and Machines* 23, 1 (2013), 105–121.

[17] Imen Khamassi, Moamar Sayed-Mouchaweh, Moez Hammami, and Khaled Ghédira. 2015. Self-adaptive windowing approach for handling complex concept drift. *Cognitive Computation* 7, 6 (2015), 772–790.

[18] B Krawczyk, LL Minku, J Gama, and J Stefanowski. 2017. Ensemble learning for data stream analysis: A survey. *Information* (2017), 1–86.

[19] Bartosz Krawczyk, Leandro L Minku, João Gama, Jerzy Stefanowski, and Michał Woźniak. 2017. Ensemble learning for data stream analysis: a survey. *Information Fusion* 37 (2017), 132–156.

[20] Luc Lévesque. 2014. Nyquist sampling theorem: understanding the illusion of a spinning wheel captured with a video camera. *Physics Education* 49, 6 (2014), 697–705. https://doi.org/10.1088/0031-9120/49/6/697

[21] Felix Mannhardt, Massimiliano de Leoni, Hajo A Reijers, and Wil MP van der Aalst. 2017. Data-Driven Process Discovery-Revealing Conditional Infrequent Behavior from Event Logs. In *International Conference on Advanced Information Systems Engineering*. Springer, 545–560.

[22] 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. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[23] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. 2000. Efficient algorithms for mining outliers from large data sets. In *ACM Sigmod Record*, Vol. 29. ACM, 427–438.

[24] Maximilian Röglinger, Johannes Seyfried, Simon Stelzl, and Michael zur Muehlen. 2017. Cognitive Computing: WhatâĂŹs in for Business Process Management? An Exploration of Use Case Ideas. (2017).

[25] Tegjyot Singh Sethi and Mehmed Kantardzic. 2017. On the reliable detection of concept drift from streaming unlabeled data. *Expert Systems with Applications* 82 (2017), 77–99.

[26] Wil MP van der Aalst. 2016. *Process mining: data science in action*. Springer.

[27] Wil M. P. van der Aalst. 2011. *Process Mining: Discovery, Conformance and Enhancement of Business Processes* (1st ed.). Springer Publishing Company, Incorporated.

[28] Robert A. Wagner and Michael J. Fischer. 1974. The String-to-String Correction Problem. *J. ACM* 21, 1 (Jan. 1974), 168–173. https://doi.org/10.1145/321796.321811

[29] Mingxi Wu and Christopher Jermaine. 2006. Outlier detection by sampling with accuracy guarantees. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 767–772.

[30] Yingxiu Zhao, Jinhai Li, Wenqi Liu, and Weihua Xu. 2017. Cognitive concept learning from incomplete information. *International Journal of Machine Learning and Cybernetics* 8, 1 (2017), 159–170.