Business process analysis based on anomaly detection in event logs: a study on an incident management case

Esther M. R. Krugger, Ana R. C. Maita, Juliana C. B. Alves, Marcelo Fantinato, Sarajane M. Peres
School of Arts, Sciences and Humanities, University of São Paulo, Brazil
{erojas, ar.cardenasmaita, m.fantinato, sarajane}@usp.br; juliana.cristina.alves@alumni.usp.br

Abstract

Business processes allow anomalies to occur during execution. Anomaly detection aims to discover behaviors that are not typical or expected in the business process. In fact, early detection helps prevent intrusion and other risks in companies. There are several approaches that address this problem in process mining. This paper discusses anomaly detection approaches in business process discovery using a real-world event log from an ITIL-covered incident management process. We discuss benefits and limitations of using knowledge from process models discovered after treating anomalies.

1. Introduction

The life cycle of business process management (BPM) involves a series of actions that are closely linked to process models, usually designed by business analysts, and data, coming from the different process-aware information systems (PAIS) that support the execution of organizational processes. According to [1], in the practice of this life cycle, process models are used in the (re)design and configuration/implementation phases, and data is used in the enactment/monitoring and diagnosis/requirements phases. However, few organizations are able to connect the information in the data to the information present in the models in order to optimize the results of the BPM. For the sake of overcoming this limitation, process mining appears as a bridge between data science and process science, enabling organizations to use information in the data (event logs) effectively to optimize process models and the BPM life cycle.

One of the most explored types of process mining today is the discovery of process models [2, 3]. The discovery techniques use event logs from PAIS and produce models without any prior information about the underlying processes. As a result, a real process model is obtained that can be used, for example, to contrast what is primarily expected from the execution of the process with what is actually implemented in the systems that support this process. Such discovery techniques assume that the event log contains a representative sample of the behavior of the process [1]. However, the event log may also contain the representation of anomalous and infrequent behaviors, which do not stand for the typical and expected behavior of the process. If this occurs, the discovered models can: *(i)* represent behavior that is not consistent with the real process [4], for instance when the anomaly comes from malicious activity or wrong implementations in the PAIS; *(ii)* represent additional behavior, whose presence in the model occurs as a side effect of trying to represent all the existing behaviors in the event log [4]; *(iii)* generate very complex process models, which are of little use to a business analyst.

The ability to detect and treat anomalies is an important feature in a process discovery algorithm, but it is still a challenge for the process mining area [1]. There are studies that aim to detect or treat anomalous paths or activities in the processes, some of which only focus on detection [5, 6, 7], others focus only on treating them [8] (i.e. repairing or filtering), and others focus in both tasks [9, 4]. They also follow different analysis principles: count-based [8], reconstruction with autoencoders-based [5, 9], prediction-based [10, 6] and knowledge-based [7]. The way anomalies are treated needs to be studied on a case-by-case basis, as, according to [10, 11], infrequent behaviors can bring important insights to the process. Thus, the focus is also to detect the anomalies in order to understand them.

Although several methods have been analyzed using real-world event logs, we have not found in the literature a comparison of methods that follow different principles applied to the same real-world event log. In addition, previous studies inserted artificial anomalies in the event logs in order to analyze the results in a supervised way, which does not represent the reality of organizations. The comparison among different methods of different principles would be relevant to provide researchers and professionals with clearer information about the benefits and limitations arising from the principles adopted in each approach. Thus, in this paper, we analyze three

URI: https://hdl.handle.net/10125/70742
978-0-9981331-4-0

anomaly detection approaches, two count-based and one reconstruction-based, using a real-world event log from an incident management system. In addition, our analysis is carried out in an unsupervised way, in order to reproduce the existing problem in organizations.

The incident management process is strongly characterized by involving entities external to the business and by involving many actions within the same life cycle step of the incident. Such characteristics represent a challenge for the exploration of the event log, as will be discussed throughout this paper. The contributions achieved with this study are: *(i)* discussion of approaches based on different principles and with application in a real case; limitations and difficulties encountered with the application of each of them are discussed based on an extensive set of experiments; *(ii)* application of the approaches in an unsupervised environment, i.e., without prior knowledge of what can be expected as anomalies; *(iii)* discussion on the benefits of discovering process models from anomaly filtering, providing useful information to the business specialist.

The remainder of this paper is organized as follows: Section 2 and Section 3 present the theoretical background and the related works, respectively; Section 4 and Section 5 aim at presenting the study setup and the discussion on the results obtained with experiments; finally, Section 6 presents the conclusion of this paper.

2. Theoretical background

2.1. Basic concepts in process mining

Process mining relies on the concepts of events, cases, traces, and logs. Given an event log L, an *event* is the occurrence of a business process activity at a given time. Event log L records events such that each event e is defined in terms of the following attributes: $\#_{id}(e)$, $\#_{timestamp}(e)$ and $\#_{activity}(e)$. A *case* corresponds to a process instance and consists of events such that each event relates exactly to a case. A case c is defined in terms of the attributes $\#_{id}(c)$ and $\#_{trace}(c)$. That *trace* is a mandatory attribute of a case and corresponds to a finite sequence of events such that each event appears only once[1]. Thus, *event log* L is a set of cases such that each event appears at most once in the full event log [1].

Process discovery consists of extracting process models from event logs, i.e., automatically producing a process model based only on an event log, using no prior information. A systematic mapping [2] shows that most researches being conducted on process mining (71%) have addressed process discovery. The predominance of process discovery is natural as other types of process mining activities usually require a process model.

[1]For analysis purposes, the traces are represented by the sequence of attribute values associated with the events that compose them. Under this representation, more than one case can follow the same trace.

2.2. Anomalies in process mining

According to [12], in data mining, anomaly detection is the process of finding data presenting behaviors that are different than expected. Anomalies are also called *outliers*, as in [13, 14]. In statistics, according to [15], an outlier is an observation that deviates so much from other observations that it arouses suspicions of having been generated using a different mechanism. Although outliers are deviations from normal behavior, they often contain useful information about atypical characteristics that affect the data generation process under analysis [13].

In process mining, an outlier is a behavior that is not or should not be part of the actual process, or an infrequent behavior [8]. In [9], outliers are seen as erroneously logged activity or resource information, or abnormal timestamps. Specific definitions of anomalies depend on the application context, and can manifest themselves as: one or more missing activities [5, 6, 8, 4], change in execution order between activities [5, 6, 8], anticipation or delay in the execution of an activity [7], re-execution of an activity [5, 6], unexpected occurrence of an exogenous or endogenous activity [7, 8, 10, 4], replacement of one activity with another [9], execution of an activity associated with an incorrect resource [5, 6], unexpected duration for activities [9, 7], long-term dependencies between activities and resources [5]. The study of anomalies in process mining is also oriented to the level of granularity in which an anomaly is sought to be identified, i.e., anomaly detection can be addressed at the level of: anomalous attributes [9, 5, 6, 4], anomalous events [5, 4] or anomalous cases [7, 5, 6].

The proposal of algorithms for anomaly detection presupposes as input a case or elements of a case (such as a window formed by events of a case). Their outputs are formatted as binary outputs or scores indicating the degree of abnormality [12, 16]. In this paper, we explore anomaly detection through two approaches proposed in process mining field: a subsequence's context-based approach (count-based) [8] and a neural-based (reconstruction with autoencoders-based) [6]. Additionally, a low frequency filtering heuristic (count-based) was explored to analyze whether the simple removal of infrequent traces could improve the process model discovery results. Those approaches are explained below.

Subsequence's context-based approach. Study [8] proposes an event log repair approach based on identifying anomalous subsequences. In this paper, we only describe the anomaly identification approach introduced

by them. Authors defined: *(i)* given a subsequence β in a specific sequence S, a context $con(\beta, S)$ is defined as the surrounding activities of a certain subsequence β in S. For instance, given a case (a, b, c), subsequence (b) is surrounded by context (a, c); *(ii)* given a context (α_1, α_2), $cov(S, \alpha_1, \alpha_2)$ of that context is defined as a subsequence of activities being covered by that context in sequence S. Thus, in previous example, the *cov* of context (a, c) is (b). Covering probability $CP(\beta, \alpha_1, \alpha_2)$ is defined as the conditional probability of β being covered by context (α_1, α_2). This probability is defined as $\frac{N}{D}$, in which N consists in the number of times that subsequence β occurs within the (α_1, α_2) context, and D consists of the total number of times the context covers subsequences with length $\leq K$. A context is significant if its frequency in the event log is higher than a threshold τ. The anomaly detection approach is defined by these parameters: K, C_L, τ, L, being K the maximum length of covered subsequences, C_L the context's subsequence lengths, τ the threshold and L the event log. Thus, given a case and a sequence of events S in that case, whose context is (α_1, α_2) and the covered sequence β, β is anomalous in that context if when tested on the following conditions returns true: *(i)* the context (α_1, α_2) is significant; *(ii)* the conditional probability CP (β, α_1, α_2) is $< \tau$. Both conditions imply counting occurrences of activity sequences, so we consider this a count-based approach.

Neural-based approach. In [5] a *denoising autoencoder* was proposed to address the anomaly detection task for both of cases and activities of an event log. An autoencoder is a type of neural network that, given an input, expects an output that is a reproduction of the input. First, authors transformed each case in the event log into a one-hot-encoding representation, including both activities and event users. In this format, the order of events is relevant. Each event is represented by a one-hot vector with dimension n. Thus, the case is represented by sequential union of event vectors. The autoenconder was trained using the backpropagation algorithm. Both input and expected output of the autoencoder have the same one-hot-encoding representation. After training, anomalous cases were expected to be reproduced with a bigger error than normal cases. Thus, a case is anomalous if reproduction error of that case is greater than an threshold (τ) or normal otherwise. τ is calculated using a scale factor α applied to the average reproduction error of the training dataset. Although it is an unsupervised anomaly detection approach, authors evaluated it in a supervised manner, as they had labels for anomalous cases.

Low frequency filtering heuristic. This approach removes cases in the event log, based on the frequency of traces (i.e., number of cases that reproduce the same trace in the event log). Given n as input parameter, removal starts with cases that reproduce the least frequent trace until it reaches n cases removed or more. More cases might be removed because more than one case can reproduce the same infrequent trace.

3. Related work

In the task of detecting anomalies in event logs, several approaches have been proposed [5, 6, 7]. In [5] the authors proposed DAE, a model based on autoencoders for detecting anomalies in event logs. They compared their results with those obtained by using other techniques usually applied in anomaly detection, showing that better results were obtained using autoencoders. However, such study, as [6, 7], is focused on assessing anomaly detection and not on how the approach can improve process discovery. One of our interests herein is the second type of perspective, since treating anomalies in event logs can improve process discovery [8].

Some authors [9, 6, 17] addressed the detection or treatment of anomalies in incident management event logs, but without an in-depth analysis of the impact of their approaches on discovering process models. In [9], the authors addressed anomaly detection and reconstruction in event logs, exploring only approaches based on autoencoders. The approach is good for reconstructing the event log after deleting anomalous behavior. Although one of the event logs analyzed is an incident management event log (BPIC 2013), they do not offer detailed assessment of their approach from the perspective of process discovery. On the other hand, [6, 17] proposed approaches based on Gated Recurrent Units for detecting anomalies in cases, events and attributes. The authors tested their technique on real-world event logs, including the BPIC 2013 event log, but evaluating the approach from the anomaly detection perspective only.

In [8] the authors demonstrated repairing anomalies can help improve the quality of the discovered processes. In that study, anomalous behavior was identified based on the frequency of activities in a context. The approach was evaluated on event logs of synthetic and real-world events, including one for incident management, BPIC2013. The authors also compare their approach with a filtering technique (Filter Matrix) and with no filtering at all. However, they did not present a comparison with approaches based on neural networks.

The comparison between neural-based and count-based approaches is necessary because approaches based on neural networks, especially autoencoders, have demonstrated a good capability to detect anomalies in business process event logs [5, 9]. On the other hand, the

count-based approach in [8] has demonstrated a good capability to improve the process discovery results after repairing the anomalous behavior. However, they both have not been compared in the same scenarios. In this paper, such a comparison is conducted in a realistic unsupervised perspective, by considering a real-world event log without labels indicating anomalies.

4. Study setting

Operational areas in organizations are constantly looking for ways to optimize their processes. This need arises mainly from processes requiring resources from several sectors of the organization. Such collaborative work makes processes complex, unstable and unpredictable. In Information Technology (IT), process optimization is sought by adopting best practices frameworks such as the Information Technology Infrastructure Library (ITIL) [18]. Among processes covered by ITIL, the incident management process addresses actions required to correct failures and restore the normal operation of a service as soon as possible, minimizing the impact on business operations [18]. In this paper, we discuss the exploration of an event log extracted from an incident management platform, used by a real-world organization, and follow the steps shown in Figure 1.

This study was carried out on an event log from an incident management process [19]. The event log was extracted from an instance of the $ServiceNow^{TM}$ platform used by an IT company, where audit data and transactional data are stored. Thus, the event log contains information regarding the context in which incidents occur and it is organized from the perspective of the incidents' life-cycle. Only information related to the state of the incidents was submitted to anomaly detection approaches.

We applied the anomaly detection approaches to detect anomalous cases and exclude them from the event log, obtaining a filtered event log. Three anomaly detection approaches were applied: a low frequency filtering heuristic; subsequence's context-based analysis [8]; neural-based analysis [5]. A procedure for excluding random cases was applied to verify whether the previous approaches are not adopting a random behavior.

All filtered event logs were input to the process discovery algorithm Inductive Miner Directly Follows (IMDF) [20]. We used a specific implementation (IMDFc) available in the PM4Py library [21]. The obtained models were evaluated under the perspective of process quality measures (fitness and precision) and in relation to the number of cases pointed out as anomalies. These quality measures were obtained by analyzing the process models against the filtered log (log used in model discovery) and against the original log. Besides evaluating whether the discovered models adequately reflect the behaviors underlying the filtered logs, this study aims to identify to what extent the models can be used as a source of information about the behaviors underlying the original log. The evaluation also includes the analysis of the parameters used in the anomaly detection approaches. Finally, we discuss the highest quality process models concerning the knowledge they can provide about the incident management process.

4.1. Event log

The data available in the event log consists of: the control flow of an incident's life cycle (e.g., status, accounting for reopenings and reassociations); incident qualifications (e.g., priority, impact, urgency, service level agreement, requesting agent, contacted agent); time attributes associated with incidents (e.g., creation date, closure date); attributes describing the assets that an incident refers to (hardware, software and services); resources involved in the handling of incidents (people and teams); and textual descriptions (clarifications about the incident, its object and how it was dealt with).

The publicly available event log[2] is anonymized, includes only part of the information available in the system and was created as a non-standardized (flat) file. It is considered an enriched event log due to the large number of attributes associated with the incidents. Thus, its contents explicitly reflect part of the information that describes the company's incident process, and implicitly represents the existence of different behaviors in the process, since it has events with the same value in the attributes, except for the timestamp attribute. As a side effect, the event log contains many loops, which makes analysis difficult. The event log has 12 months (Mar-2016 to Feb-2017) of activity records, totaling 24,918 traces and 141,712 events. Figure 2 shows some attributes of a case present in the enriched event log. Colored highlights indicate the information present in the simplified event log, some process characteristics and conditions that might influence the anomalies detection results. Such characteristics are indirectly represented in the simplified event log.

For execution of this study, only the information about the case identifier, the state of the incident and the timestamp were presented to the algorithms. Hatched area in Figure 2 composes the information in the simplified event log. Anomaly detection approaches and algorithms for process models discovery and extracting quality measures have, depending on the event log conditions, a high computational cost. In addition, even if

[2]Details about the event log can be found at `https://archive.ics.uci.edu/ml/datasets`

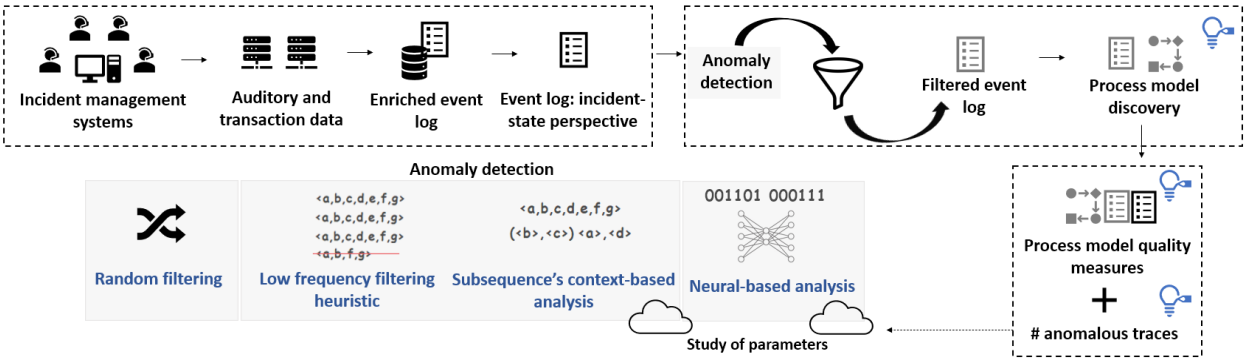

Figure 1. Method applied in this study

event id	case id	incident-state	sys_updated_at	sys_updated_by	category	group_ assignment	assigned_to
1	INC 131	Active	02/03/2016 13:41	Updated by 495	Category 45	Group 10	Resolver 139
2	INC 131	Resolved	06/04/2016 14:29	Updated by 495	Category 45	Group 10	Resolver 139
3	INC 131	Active	06/04/2016 16:31	Updated by 351	Category 45	Group 10	Resolver 139
4	INC 131	Active	06/04/2016 16:31	Updated by 351	Category 45	Group 10	Resolver 139
5	INC 131	Awaiting User Info	14/04/2016 09:57	Updated by 495	Category 45	Group 10	Resolver 139
6	INC 131	Awaiting User Info	25/04/2016 10:50	Updated by 495	Category 42	Group 39	?
7	INC 131	Resolved	25/04/2016 11:16	Updated by 748	Category 42	Group 39	Resolver 194
8	INC 131	Closed	30/04/2016 12:07	Updated by 908	Category 42	Group 39	Resolver 194

Data on the event log used in the study

Category change | Group reassignment | Resolver reassignment | Incident reactivation | Missing data

Figure 2. Excerpt from the event log

the execution cost was feasible, the complexity of the process models when various descriptive attributes of the incident are considered makes the model interpretation and qualitative analysis by business analysts very difficult and sometimes impossible. The choosing of these attributes is also capable of implicitly representing changes in the value of other attributes of the process. The occurrence of these changes is represented by the loops created in the incident state and by the performance information (timestamp). In this study, performance information is not being addressed. Timestamp is used only to sort the events of each case.

4.2. Process model discovery: strategy and quality measure

The IMDF is state of the art regarding process model discovery from event logs. This algorithm is an adaptation of the Inductive Miner framework (IM) [22] that works on directly follows graphs rather than on event logs. The computation performed by this adaptation can be parallelized and the size of the resulting graph depends quadratically on the number of activities in the event log. The number of traces or events in the event log does not impact the size of the generated graph. The version of this algorithm implemented for PM4Py, used herein, is optimized to build a sound process model, with good values of fitness (in most cases, the algorithm guarantees maximum fitness) [21]. Such an implementation receives an event log as an input and produces a process model as an output, which is represented either as a Petri net or as a process tree. In order to facilitate understanding, in this paper, the generated models were converted to BPMN notation. Figure 3 shows the process model discovered for the incident event log, considering the state of the incident perspective (attribute *incident_state*). All gateways in Figure 3 refer to *split* or *join XORs*, whose existences were discovered from the event log. These gateways show the activities are both optional and can be carried out within loops.

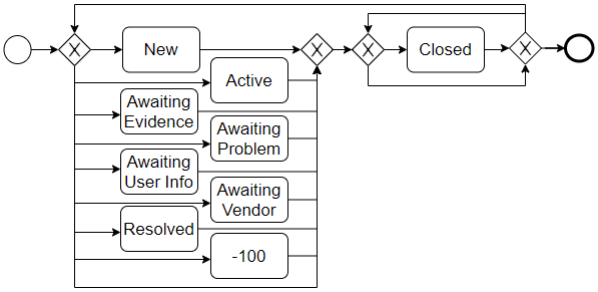

Figure 3. Process model obtained for the event log with the incident's state perspective

In order to evaluate discovered models, we are using two process model quality measures [1, 23]. *(i) Fitness* (or *completeness*) (F): assesses whether a process model can reproduce the behavior present in the event log. A process model is fully complete if all traces in the event log can be replayed by the model from beginning to end; *(ii) Precision* (P): assesses whether a process model can generate only the behavior contained in the event log. A process model is fully precise if every trace produced by the model is contained in the event log.

According to [23], a highly *complete* process model is not useful if it is too *imprecise*. Thus, these authors claim that an algorithm that discovers process models

should balance *fitness* and *precision*. They used the *F-score* (FS) calculated with *fitness* and *precision* as: $FS = 2 * (\frac{F*P}{F+P})$. These measures were applied in this study based on two quality analysis strategies:

1^{st} quality analysis strategy. In this procedure, the assessment was mainly based on fitness and F-score measures. We looked for models that achieved at least 0.95 in fitness. Thus, models that were unable to reproduce a significant part of the cases in the event logs were not considered reliable to reveal knowledge on the process' behavior. Then, models that did not reach values of at least 0.9 in F-score over the original event log were discarded, so that the remaining models presented a good balance between fitness and precision. However, high fitness values in the quality measure of the model over the original event log can be a side effect of low precision of the model over the filtered event log (low precision means there are traces in the process model that do not exist in the event log from which the model was generated). To avoid this situation, we also discard all models that showed a high difference ($>= 0.25$) between the precision values obtained on the original event log and the filtered event log.

2^{nd} quality analysis strategy. This procedure chooses the best results based on fitness and precision measures. We discarded models that did not reach the minimum value of 0.9 in both measures, against both event logs. This is a more restrictive analysis. Although the fitness values were relaxed compared to the first procedure's lower limit, the precision values must be high. A smaller number of models is selected under this procedure.

4.3. Parameter setup

The use of *low frequency filtering heuristic* requires one parameter: the number of cases to be excluded (n). We performed this heuristic with n between 1,000 and 24,000, with steps of 1,000 cases. We got 20 process models after filtering infrequent cases. We did not get 24 process models because some parameter values filter the same cases and produce the same process models.

The application of *subsequence's context-based approach* requires the following parameters to be defined: subsequence length, threshold and context's subsequence length. The third parameter was fixed at 1 (C_L = 1). The strategy was executed 150 times, from a grid with the following combinations of values for the parameters: maximum length of covered subsequences K: $\{1, 2, 3, 4, 5\}$; threshold τ: $\{0, 0.1, 0.2..., 0.9, 1\}$, $\{0.01, 0.02, ..., 0.09, 0.11\}$ and $\{0.91, 0.92..., 0.99\}$. Thus, after filtering cases detected as anomalous and discovering process models for each of them, we obtained 150 process models.

The *neural-based approach* is based in denoising autoencoder proposed in [6]. We implemented a similar input representation (one-hot encoding) and architecture. Main differences between our parameter settings and those proposed by [6] are: (i) we trained denoising autoencoder using batchs of size 500; (ii) we do not use validation set to make feasible to compare denoising autoencoder with subsequence's context-based approach. Since subsequence's context-based analysis use the whole event log to detect anomalies, we also used the whole event log to train the denoising autoencoder. We defined an early stopping when training loss is increasing in last 10 epochs and defined a maximum number of training epochs in 300; (iii) denoising autoencoders consist of an input layer with 522 neurons and two hidden layers settled to be half the input layer size (261 neurons). Such parameters were settled as fixed, as well as activation function for output layer (function G = {sigmoid}), thus we focused in exploring other three parameters: activation functions for the two hidden layers (function F), learning rate (α), and a scaling factor (σ). This approach was executed 270 times, from a grid with the following combinations of values for the parameters: function F = {relu, sigmoid, tangent hyperbolic}; α = {0.001, 0.5, 1}; σ = $\{0, 0.1, 0.2..., 0.9, 1\}$, $\{0.01, 0.02, ..., 0.09, 0.11\}$ and $\{0.91, 0.92..., 0.98, 0.99\}$. Thus, after filtering anomalous cases for each execution, we obtained 270 filtered event logs and the same number of process models.

5. Results and discussions

This section presents discussions on the results obtained in this study. First, we developed an analysis of the relationship between the parameters used in the anomaly detection approaches and the quality measures of the process models. Next, we perform a qualitative analysis on the selected process models. We compared the anomaly detection approaches for characterizing what each one considered as anomalies, and highlighted the knowledge that can be extracted from event logs without anomalies in the context of the incident management process. For such analyzes, we considered the set of models selected with the 1^{st} and 2^{nd} quality analysis strategy. The 2^{nd} quality analysis strategy selects a few models, so we only present the quality measures for them.

5.1. Parameters analysis and quality measures

In this section, the results selected to observe the behavior of parameters and the quality measures followed the 1^{st} quality analysis strategy, but without the restric-

tion on the F-score measure. Thus, eight process models remained for low frequency filtering heuristic, 39 process models for subsequence's context-based approach and 143 process models for neural-based approach[3].

Low frequency filtering heuristic. The results considered in this approach have filtered up to 8,107 anomalous cases. Analysis of the data revealed that the elimination of some anomalous cases (less than 16%) generates models with low precision both in the filtered event log and in the original event log. On the other hand, the elimination of more than 20% cases generates models in which the precision on the filtered event log decreases, although the precision for the original event log is high and close to 1. The models generated by this approach reach maximum precision and fitness (greater than 0.936), both on the filtered event log and on the original event log.

As mentioned above, if less than 16% and more than 20% anomalous cases are eliminated, the generated models show a drop in the precision on the filtered out event log. Therefore, these dropped cases are causing the model to be imprecise regarding the behavior of the filtered event log, while remaining precise relative to the original event log. Models are likely to allow additional behavior than what exist in the filtered event log, so these behaviors could be contributing to the precision of the original event log.

In that sense, we demonstrate the ability to generate models with high quality from the removal of anomalies (or infrequent behavior). These models were able to correctly represent the real behavior seen in the original event log and at the same time to correctly represent the main behavior (or frequent behavior) seen in the filtered event log. However, determining the appropriate number of cases to be eliminated is dependent on an exhaustive search of the entire parameter space.

Subsequence's context approach. The analysis of the selected models allowed to identify behavior patterns of the quality measures of process models and the number of cases pointed out as anomalous (see Figure 4).

The content in Figure 4 shows that subsequences of length 4 and 5 led to the detection of anomalies that, frequently, did not allow the obtaining of models which met the quality selection criteria. However, this subsequence size allowed the achievement of the two best filters obtained in this anomaly detection approach (see Table 1). The results obtained with smaller subsequences lengths lead to the generation of models with low precision on the filtered event log, showing that some complexity still

[3]In addition to the three approaches discussed herein, random filtering was carried out to attest that random solutions could not solve the anomaly detection problem. No results obtained with random filtering achieved the restrictions of the quality analysis strategies.

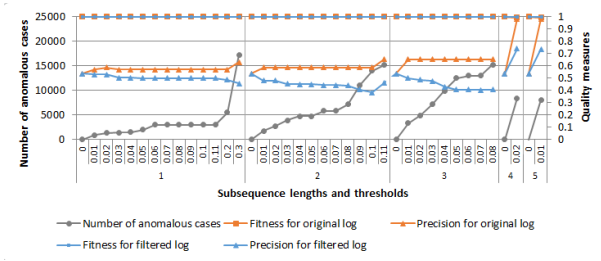

Figure 4. Parameters behavior for the subsequence's context approach

remains in the event log even with the removal of the pointed anomalies. The process model discovery algorithm, when meeting specific traces still present in the event log, generates a variety of transitions in the model making it imprecise. Subsequences of size 2 achieve better results by eliminating a larger number of cases (between 10,000 and 15,000 cases). The subsequences with size 4 and 5 eliminate fewer cases (around 8,000) and achieve the best results. Regarding the thresholds, the predominance of low value thresholds (less than or equal to 0.3) is observed. In general, large subsequences generate good results combined with very small thresholds. Smaller subsequences yield your best results with thresholds close to 0.01.

Neural-based analysis. Following the same quality analysis strategy for selecting models, for this approach it was also possible to observe some behavior patterns for the parameter values (see Figure 5). Learning rate and scaling factor influenced the filtering results in this strategy. The different activation functions used in the hidden layers of autoencoder did not influence the quality of the results. Scaling factors greater than 0.9 resulted in fewer cases being filtered and had stable and low fitness and precision measures. For cases with higher learning rates $(0.5, 1)$, low scaling factors provide improved precision for the original event log but worsened precision for the filtered event log. As an effect, the high fitness value for the original event log may be unwanted. However, the combination of scaling factors between 0.6 and 0.9 and a learning rate of 0.001 creates the right conditions for good filtering. In a macro view, the values of the learning rate also do not generate large variations in results, except when combined with scaling factors between 0.6 and 0.9.

Table 1 presents details of quality measures related to process model for best results obtained using the two quality analysis strategies. Using the 1^{st} quality strategy, five process models remained for low frequency filtering heuristic, two process models for subsequence's context-based approach and four process models for

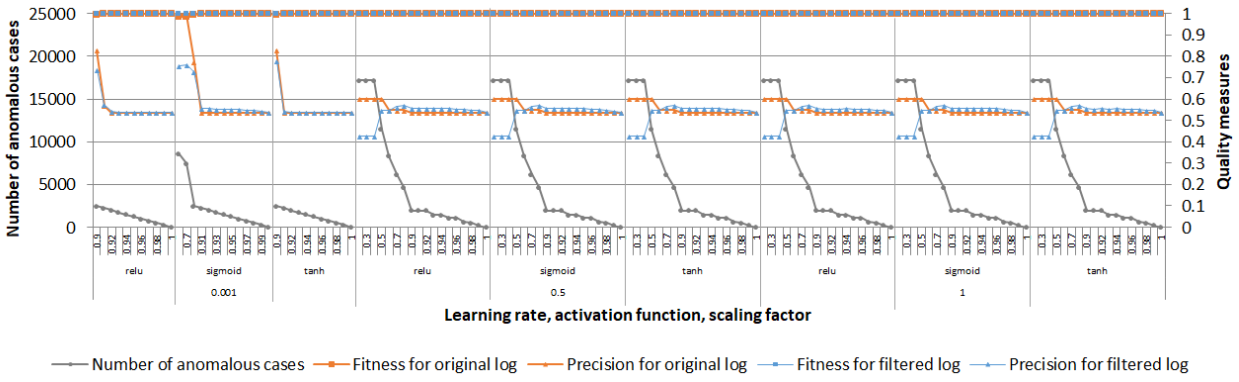

Figure 5. Parameters behavior for the neural-based analysis

neural-based approach. Using the 2^{nd} quality strategy, seven process models remained for first approach, one process model for the second and one process model for the third approaches. The low frequency filtering heuristic generated some results with equal values for the process model quality measures. In this case, only one of the models is listed in the table. Quality measures for the process model obtained using the original event log (without filtering) are also presented.

5.2. Process models analysis

In this section, we present the analysis of the process models discovered from the original and filtered event logs, considering the results listed in Table 1. Figure 3 shows the process model discovered from the original event log. This model contains all activities present in the event log, including an activity exogenous to the business process (*-100*). The model has a high fitness, but low precision. Many extra behaviors are allowed because nine activities are connected to each other through a loop. Moreover, the model allows for empty traces, which does not make sense from a business perspective and does not agree with good process modeling practices. Although this model reflects the event log, it does not provide really useful knowledge, as it does not abstract a structure for the business process.

The anomaly detection approaches filter the original event log in different ways, pointing out more or less anomalous cases, according to the combination of parameters applied. Some identical models were generated by more than one approach or by the same approach applied with different parameter values. All models obtained in the filterings under analysis are shown in the Figures 6, 7 and 8, with reference to the models selected with the 1^{st} and the 2^{nd} quality measure strategies. None of models generated from filtered event logs produce an empty trace. A summary of the main characteristics related to each model is presented in Table 2.

The filtered event logs allowed generating process models in which it is possible to observe behaviors similar to the behaviors expected for an incident management process. Thus, for extracting knowledge about the business process, these models are more useful than the model in Figure 3. The process models C/D showed in Figure 6 reveal the following characteristics for the business process underlying to the event log: the process starts with the activity *new*; despite the existence of loops in each activity (due to updates to context attributes - see Section 2), there is a sequence of activities (*new*, *active*, *awaiting user info*, *resolved*, *closed*) executed in the cases of the filtered event log which is predominant in the original event log.

In general, the three approaches considered infrequent activities as anomalous. The noisy activity *-100* was eliminated in all models. The (expensive) waiting activities (*awaiting evidence/problem/vendor/user info*) were eliminated in one or more models. Eliminating these activities is not necessarily beneficial. On the one hand, the models show that waiting for user's information is part of the mainstream of the process, alerting the business analyst. On the other hand, the other waiting activities are hidden from the analyst, generating misinformation. A similar reasoning can be outlined for the treatment of cases that present loops. On the one hand, knowing that there is a loop in the process can be important to trigger optimization actions in the organization. On the other hand, inserting infrequent loops into the model causes the model's precision to deteriorate, leading to false conclusions about the business process. Such situations are inherent to the study of anomalies, so it may be interesting for the business analyst to access a process model generated from information about the traces associated with anomalous cases.

Table 1. Quality measures of process models for best results

Measures for full event log model: fitness (F) = 1.000, precision (P) = 0.536 and F-score (FS) = 0.698																				
Subsequence's context-based							**Neural-based**							**Low frequency filtering heuristic**						
anom. cases	F_f	P_f	FS_f	F_o	P_o	FS_o	anom. cases	F_f	P_f	FS_f	F_o	P_o	FS_o	anom. cases	F_f	P_f	FS_f	F_o	P_o	FS_o
1^{st} quality analysis strategy																				
							2,485	1.000	0.735	0.847	0.995	0.829	0.904							
							2,491	1.000	0.781	0.877	0.995	0.829	0.904							
														4,006	1.000	0.936	0.967	0.989	0.988	0.989
							7,427	1.000	0.760	0.863	0.987	0.996	0.991	7,034	1.000	0.865	0.928	0.987	0.996	0.991
8,074	1.000	0.738	0.849	0.992	0.982	0.987	8,526	1.000	0.753	0.859	0.987	0.996	0.991	8,107	1.000	0.884	0.938	0.950	0.999	0.974
8,450	1.000	0.740	0.851	0.992	0.982	0.987														
2^{nd} quality analysis strategy																				
							4,013	0.916	0.950	0.933	0.919	0.973	0.945	4,006	1.000	0.936	0.967	0.989	0.988	0.989
10,252	1.000	0.975	0.987	0.942	0.999	0.969								10,252	1.000	0.975	0.987	0.942	0.999	0.969
														13,039	1.000	0.936	0.967	0.942	0.999	0.969

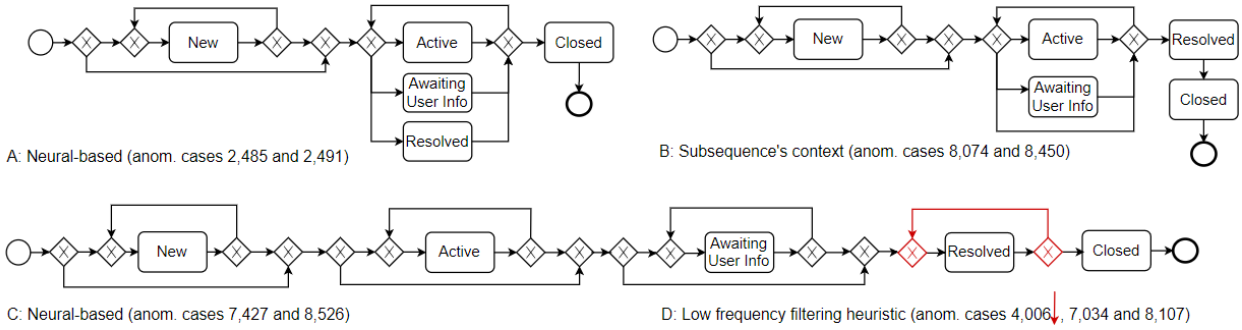

Figure 6. Process models selected by the 1^{st} quality measure strategy

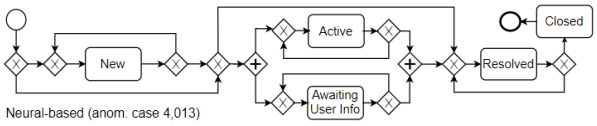

Figure 7. Process model selected by the 2^{nd} quality measure strategy

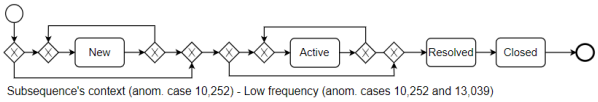

Figure 8. Process model selected by the 2^{nd} quality measure strategy

6. Conclusion

In this paper, we analysed the ability of three anomaly detection approaches to improve discovering process model task, by filtering anomalous cases in a real-world event log. We perform both quantitative and qualitative analysis in an unsupervised setting. There is not evidence of such approaches were compared previously in literature despite of being suitable candidates for improving process discovery. In this paper we explored which parameter values influenced the most in quality models. Also, we were able to find some behaviour patterns in parameter values. We observed that the neural-based approach was the only one capable of detecting a low number of anomalous traces (about 2,000) that allowed to generate a filtered event log from which process models with good quality were discovered. This shows that the autoencoder is capable of detecting the anomalies that most affect model discovery. For runs that filtered about 4,000 cases, the neuron-based approach and low-frequency filtering heuristics are competitive. However, the former led to the discovery of a process model with parallelism (Figure 7), which may show more effectiveness in reducing the variety of anomalous behavior in the event log.

Our study is limited to a single event log with unique characteristics. Thus, the results presented herein do not allow extrapolating the findings to other event logs or business domains. Thus, one way to expand this research includes the design of a systematic experimentation setting in which it is possible to produce results that allow statistical generalization. In addition, the involvement of stakeholders from the business area could bring more expressive insights to qualitative analysis.

Table 2. Qualitative analysis of process models for best results. Columns: strategy used to obtain model; possible start activities; possible end activities; activities not included in model; loops not included in model; percentage of anomalous cases removed from the event log prior to model discovery

Strategy	Start activities	End Act.	Anomalous activities	Anomalous loops	%Anomalous cases
No filtering	Any	Any	–	All activities are involved in loops	–
			1^{st} quality analysis strategy		
Seq 8,074 / Seq 8,450	A, AUI, N, R	C	-100, AE, AP, AV	C, R	32 (Seq 8,074) / 34 (Seq 8,450)
Neu 2,485 / Neu 2,491	A, AUI, N, R	C	-100, AE, AP, AV	C	10 (Neu 2,485) / 10 (Neu 2,491)
Neu 7,427 / Neu 8,526	A, AUI, N, R	C	-100, AE, AP, AV	C, R	30 (Neu 7,427) / 34 (Neu 8,526)
Freq 4,006	A, AUI, N, R	C	-100, AE, AP, AV	C	16
Freq 7,034 / Freq 8,107	A, AUI, N, R	C	-100, AE, AP, AV	C, R	28 (Freq 7,034) / 33 (Freq 8,107)
			2^{nd} quality analysis strategy		
Seq & Freq 10,252 / Freq 13,039	A, N, R	C	-100, AE, AUI, AP, AV	C, R	41 (Seq & Freq 10,252) / 52 (Freq 13,039)
Neu 4,013 / Freq 4,006	A, AUI, N, R	C	-100, AE, AP, AV	C	16 (Neu 4,013) / 16 (Freq 4,006)

7. Acknowledgments

This study was financed in part by the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior* - Brazil (CAPES) - Finance Code 001

References

[1] W. Van der Aalst, *Process mining: Data science in action*. Springer Berlin Heidelberg, 2 ed., 2016.

[2] A. R. C. Maita, L. C. Martins, C. R. L. Paz, L. Rafferty, P. C. K. Hung, S. M. Peres, and M. Fantinato, "A systematic mapping study of process mining," *Enterp. Inf. Syst.*, pp. 1–45, 2017.

[3] C. S. Garcia, A. Meincheim, E. R. F. Junior], M. R. Dallagassa, D. M. V. Sato, D. R. Carvalho, E. A. P. Santos, and E. E. Scalabrin, "Process mining techniques and applications – a systematic mapping study," *Exp. Syst. Appl.*, vol. 133, pp. 260 – 295, 2019.

[4] R. Conforti, M. L. Rosa, and A. H. M. t. Hofstede, "Filtering out infrequent behavior from business process event logs," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 2, pp. 300–314, 2017.

[5] T. Nolle, S. Luettgen, A. Seeliger, and M. Mühlhäuser, "Analyzing business process anomalies using autoencoders," *Mach. Learn.*, vol. 107, no. 11, pp. 1875–1893, 2018.

[6] T. Nolle, A. Seeliger, and M. Mühlhäuser, "Binet: Multivariate business process anomaly detection using deep learning," in *Bus. Process Manage.*, pp. 271–287, 2018.

[7] S. Pauwels and T. Calders, "An anomaly detection technique for business processes based on extended dynamic bayesian networks," in *Proc. of the 34th Symp. on Appl. Comput.*, pp. 494–501, 2019.

[8] M. Fani Sani, S. J. van Zelst, and W. M. P. van der Aalst, "Repairing outlier behaviour in event logs," in *Bus. Inf. Syst.*, pp. 115–131, 2018.

[9] H. T. C. Nguyen, S. Lee, J. Kim, J. Ko, and M. Comuzzi, "Autoencoders for improving quality of process event logs," *Exp. Syst. Appl.*, vol. 131, pp. 132 – 147, 2019.

[10] F. Mannhardt, M. de Leoni, H. A. Reijers, and W. M. P. van der Aalst, "Data-driven process discovery - revealing conditional infrequent behavior from event logs," in *Adv. Inf. Syst. Eng.*, pp. 545–560, 2017.

[11] M. Werner and M. Nüttgens, "Improving structure: Logical sequencing of mined process models," in *Proc. of the Annual Hawaii Int'l Conf. on System Sciences*, pp. 3888–3897, 2014.

[12] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*. Elsevier Inc., 3 ed., 2012.

[13] C. C. Aggarwal, *Outlier Analysis*. Springer-Verlag New York, 2 ed., 2016.

[14] H. Wang, M. Bah, and M. Hammad, "Progress in outlier detection techniques: A survey," *IEEE Access*, vol. 7, pp. 107964–108000, 2019.

[15] D. Hawkins, *Identification of Outliers*. Springer Netherlands, 1 ed., 1980.

[16] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, pp. 15:1–15:58, July 2009.

[17] T. Nolle, S. Luettgen, A. Seeliger, and M. Mühlhäuser, "Binet: Multi-perspective business process anomaly classification," *Inf. Syst.*, 2019.

[18] itSMF, "Global survey on IT service management." The IT Service Management Forum, 2013.

[19] C. A. L. do Amaral, M. Fantinato, and S. M. Peres, "Attribute selection with filter and wrapper: An application on incident management process," in *Proc. of the 2018 Federated Conf. on Comput. Sci. and Inf. Syst.*, vol. 15, pp. 679–682, 2018.

[20] S. Leemans, D. Fahland, and W. van der Aalst, "Scalable process discovery and conformance checking," *Software Syst. Model.*, no. 17, p. 599–631, 2018.

[21] A. Berti, S. J. van Zelst, and W. M. P. van der Aalst, "Process mining for Python (PM4Py): Bridging the gap between process and data science," *CoRR*, vol. abs/1905.06169, 2019.

[22] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Discovering block-structured process models from event logs - a constructive approach," in *Appl. and Theory of Petri Nets and Concurr.*, pp. 311–329, 2013.

[23] A. Augusto, R. Conforti, M. Dumas, M. L. Rosa, and A. Polyvyanyy, "Split miner: Automated discovery of accurate and simple business process models from event logs," *Knowl. Inf. Syst.*, vol. 59, no. 2, pp. 251–284, 2019.