Whetting the SWORD: Detecting Workarounds by Using Active Learning and Logistic Regression

Wouter van der Waal
Utrecht University
w.g.vanderwaal@uu.nl

Inge van de Weerd
Utrecht University
g.c.vandeweerd@uu.nl

Saskia Haitjema
University Medical Center Utrecht
S.Haitjema@umcutrecht.nl

Teus Kappen
University Medical Center Utrecht
T.Kappen@umcutrecht.nl

Hajo A. Reijers
Utrecht University
h.a.reijers@uu.nl

Abstract

In many organizations, especially in healthcare, workers may work around prescribed procedures. Detecting these workarounds can give insights into difficulties concerning the procedures, which in turn can be used to improve them. Previous studies have shown that workarounds may be discovered from an event log using a set of predefined patterns such as the duration of a trace or the number of resources involved in one. However, domain experts may find it difficult to evaluate and monitor results if there are multiple patterns that indicate workarounds. Training a model that merges the features is often difficult because there are no available datasets covering workarounds. Labeling traces generally requires a lot of time from domain experts. In addition, this would have to be repeated for every new domain, company, or even department since the types of workarounds that occur may differ strongly between them. In this work, we propose to combine the features using a Logistic Regression model and train through Active Learning. In a case study at a hospital, we find that after training the model on only 10 to 15 traces, it stabilizes with an approximate F1 score of .75. This shows that we create and train a model that can detect workarounds well without requiring a large amount of labeled data or a lot of time from a domain expert.

Keywords: Workarounds, Event log, Active Learning, Logistic Regression, Process Mining.

1. Introduction

Workarounds are deviations from the prescribed procedure to reach an intended goal (Ejnefjäll & Ågerfalk, 2019; van der Waal et al., 2022). Especially in hospital settings, they are a widespread phenomenon (Azad & King, 2008; Beerepoot et al., 2021; Röder et al., 2014; van der Waal et al., 2022), but workarounds are also commonplace in other sectors, such as retail (van de Weerd et al., 2019), or consulting agencies (Wolf & Beverungen, 2019). The discovery of workarounds can help to identify problems in processes and systems. They contain valuable information on how the process could be improved.

Workarounds may be discovered through interviews and observations, but recent work shows that they can be detected in a more efficient and automated way by using event logs that contain sequences of events, also called traces. Various studies have identified patterns that indicate that a trace may contain workaround behavior. For example, certain activities may be skipped (Outmazgin & Soffer, 2014) or repeated (Alter, 2014; Weinzierl et al., 2022), certain process participants may perform tasks they are not authorized for (Beerepoot et al., 2021; Outmazgin & Soffer, 2014), or activities are executed at unlikely times (van der Waal et al., 2022).

To classify a deviating trace as a workaround instead of user error or malicious behavior, we need to know the intent behind the diverging trace. Thus, even if patterns are used to indicate workaround candidates, these candidates need to be evaluated by an expert before workarounds can be used for process improvements (Bezemer et al., 2019). Previous work (van der Waal et al., 2023) proposes to create a ranking of the traces that result from applying a pattern, which reflects the likelihood that a trace is a workaround. This already helps to prioritize which traces should be evaluated. The idea of a ranking is that there is always one most interesting trace to evaluate. However, if there are multiple patterns taken into account, there are also multiple most interesting traces, which limits the effectiveness of the approach. Combining the patterns is not straightforward since every ranking follows a completely different distribution. Minimum and maximum scores may also differ strongly per pattern. Even when normalized between 0 and 1, means and standard deviations are rarely similar.

One way to combine patterns is to express them as scores that indicate how strong the pattern matches and use this as input features for machine learning models. However, training complex models usually requires a large dataset. Such a dataset containing many traces labeled as normative or workaround does not currently exist. Constructing one would require an expert to manually label hundreds of traces, which would not be feasible. Another challenge is that such a model would be specific to the domain and dependent on the workaround patterns used. Since workarounds can differ strongly per domain (Weinzierl et al., 2022), organization (Ejnefjäll & Ågerfalk, 2019), or even department (van der Waal et al., 2023), the model needs to be retrained from scratch for every new application, which would require a sizable time investment from interested process managers.

To address the problem of combining the patterns, as well as the data requirement and training issues that arise when we do, we propose to make efficient use of the limited time domain experts have available. We do so by using Active Learning (Settles, 2009) to train a Logistic Regression model that combines multiple features into a single score. We evaluate this approach with a case study at the emergency room of the University Medical Center Utrecht. We train the model during a single one-hour interview in which we identify several workaround types. To evaluate the performance of the model, we create a testing set by defining specific rules to detect all instances of these types in the event log. Using this testing set, we show that our approach can distinguish traces as normative or workaround after a short training session.

The primary contribution of this paper is an approach to detect workarounds in event data. It is an extension of the SWORD framework (van der Waal et al., 2022) to make it more applicable and useful. The extended approach involves Active Learning to ensure that the data requirements can be fulfilled *with limited queries for a domain expert*. By employing Logistic Regression it then becomes possible to *combine features* to establish the occurrence of a workaround from various perspectives.

From a practical point of view, the model can be quickly tailored to a new application domain, because of to the little training time required. In addition, combining features makes complex multi-feature frameworks much easier to use for real-life workaround detection by process managers. For scientific purposes, our approach requires very little time and effort from domain experts to both train and test a model. This can help future studies where no suitable datasets are available or possible to create. While our focus is on workaround detection, this approach may apply to other process mining fields where traces need to be classified by domain experts.

The remainder of the paper is structured as follows: In Section 2 we explore the current state of the art concerning workaround mining, as well as common approaches in statistics and data science to create models using limited data. In Section 3 we further explain our approach to create, train, and evaluate our model. Then in Section 4, we evaluate the performance of our model using a case study. Finally, we discuss our findings and limitations in Section 5 and conclude the paper in Section 6.

2. Related Work

2.1. Workaround Mining

There are two main goals for workaround mining: discovering new workaround types and detecting instances based on these types. Traditionally, workaround types can be discovered through qualitative measures, such as interviews and observations (Beerepoot & van de Weerd, 2018). By defining a rules, new instances of discovered type can be detected in new datasets (Beerepoot et al., 2021). In recent work, there has been an increase in attention to discovering workaround types using quantitative, data-driven methods, which we discuss here.

Outmazgin and Soffer (2014) define six patterns to mine workarounds from event logs. While the patterns are set up broadly, they require specific domain knowledge to detect in an event log. For example, the "Incompliance to role definition" pattern requires a known set of users that are allowed to perform a specific activity. As such, while this approach can find new instances of known workaround types, discovering completely new types may be difficult.

The SWORD framework makes use of 22 patterns (van der Waal et al., 2022). Depending on the data requirements of a corresponding pattern, new instances of known workaround types can be detected, or new types may be discovered. Depending on the case, some patterns may turn out to be more useful than others. This can make it challenging for domain experts to evaluate the large number of patterns.

Weinzierl et al. (2022) use neural networks to detect workarounds using seven patterns defined in various other papers. While this approach manages to reach F1 scores up to 0.83, depending on the case, it has only been tested on synthetic data for now. In addition, training a neural network usually requires a large dataset, which is not always available for workaround mining.

Starting from an event log, Wijnhoven et al. (2023) first discover a process model. This model is used to investigate 11 patterns that may indicate workarounds. While they show that this approach can be applied in practice, there is the underlying assumption that a process model is discoverable. This may not hold in complex scenarios, such as healthcare. For example, there may be theoretical procedures in the emergency room, but patients rarely follow the same path. To complicate matters more, patients may arrive with multiple ailments that are treated in parallel, making process model discovery infeasible using the limited data available.

2.2. Data Requirements

Existing workaround detection tends to rely on readily labeled data, where the labels indicate the specific workarounds. When only very limited labeled data is available, common techniques such as Support Vector Machines or Decision Trees perform poorly (Sordo & Zeng, 2005) or require users to manually inspect many patterns and instances to identify workarounds. Active learning (Schein & Ungar, 2007; Settles, 2009) is a stream of algorithms in machine learning that can interactively query a user to optimize the learning rate and improve performance. As a result, Active Learning can handle initially unlabeled data and minimize the user's effort. Active Learning aims to do this by predicting the information gain for every possible query. Selecting the option with the highest expected gain would then lead to training a strong model using limited data. Since expected information gain must be determined, Active Learning does restrict the possible models that may be used with it. Two common, straightforward classification models that have been used with Active Learning are Naive Bayes and Logistic Regression.

The Naive Bayes classifiers (Lewis, 1998) use probabilistic reasoning to classify items by predicting the highest chance label based on a set of features as input. While they can reach high accuracy levels, they assume the individual features are independent. However, many of the features currently used in workaround mining are strongly related. For example, one indication of a workaround is the number of process participants involved in a trace. This may initially seem unrelated to another pattern that relates to the total duration of the trace. However, if a trace takes a long enough time, it may be natural that more participants start being involved because of, for example, shift changes.

Logistic regression (Kleinbaum & Klein, 2002) is a regression technique that assigns a score between 0 and 1 to a set of features as input. Binary classification is simply done by setting a cut-off value. Any result above this value would be considered one option (e.g., a workaround), and the ones below it the other (e.g., a normative trace). The model is trained by adjusting the weight of each feature, making certain patterns either more or less important than the others to classify traces as workarounds. As opposed to the Naive Bayes classifier, Logistic Regression has no issues with strongly related features, making it well suited for workaround mining. Thus, for this study, we will use Logistic Regression as the model and train it through Active Learning.

3. Approach

In this section, we explain the proposed approach to combine multiple workaround patterns into a single score. As shown in Figure 1, we start by describing how we construct the Logistic Regression model. Then, we discuss how we train this with domain experts using Active Learning. Finally, we talk about how we can evaluate the performance of the model.

3.1. Constructing the Model

To construct our Logistic Regression model, we start with an event log. As we discussed in Section 2, we use the SWORD framework (van der Waal et al., 2022) to assign features to individual traces from such a log in this paper. Table 1 shows two example patterns for each perspective from the full framework.

All 22 patterns in the SWORD framework have separate data requirements, meaning that depending on the data available in the event log, specific patterns may be applied. For example, activities executed by a different-than-usual resource, i.e., a process participant, can indicate a workaround, but to detect it, information about resources must be in the event log. Based on these requirements, we select the patterns that can be used for the specific log. When applied, each pattern leads to one score per trace that reflects its deviation from the norm as established by the full set of traces. While these scores were previously used to create a ranking per pattern, in this study, we use the scores as input for the model.

As discussed in Section 2, we use Logistic Regression for actively detecting workarounds, since the features from the SWORD framework are not independent. As illustrated in Table 2, each trace and pattern is assigned a score. The set of scores for all traces and patterns is the input of the model. After training, each trace has a value between 0 and 1. When a trace is assigned a value close to 0, this indicates there is likely no workaround; when the value is close to 1, there is very likely a workaround. Since this is binary classification, the default cutoff point is 0.5. If a score is lower than 0.5, the trace will be

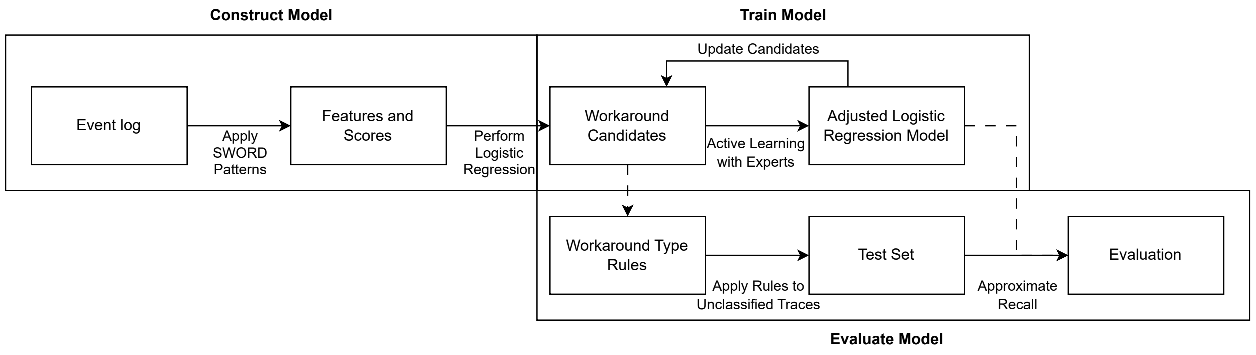

Figure 1. A schematic overview of how we create, train, and evaluate a model starting from an event log.

classified as normative, otherwise as a workaround.

3.2. Training the Model

To optimize the limited data we can query from domain experts, we use Active Learning to train the model. Active Learning aims to optimize the training speed of a model by selecting the item that will most likely provide the highest information gain (Schein & Ungar, 2007). Depending on the model, the selection process may be very different. In the case of binary classification, the item with the highest uncertainty generally provides the most information gain (Settles, 2009). Since Logistic Regression assigns a score between 0 and 1 to items, continuously selecting the item that is closest to 0.5 will likely train the model fastest (Lewis & Catlett, 1994).

To train the Logistic Regression model, we interviewed a domain expert. In doing so, we present the trace with the highest expected information gain and ask whether it should be classified as normative or a workaround. Since the model is relatively simple, we can immediately use this classification to retrain the model. We then present the trace that offers the highest gain for this new model and ask the same question. We continue this process until the expert is out of time. Note that if multiple traces provide the same highest expected information gain, such as at the start when all traces are equally uncertain, we select one randomly.

3.3. Evaluating the Model

Common measures to quantify the performance of classification models are recall and precision (Powers, 2011). Recall compares the true positives and the false negatives to discover how many of the relevant items are found. Precision compares the number of true positive and false positive classifications to determine how many of the matches are relevant in the model. For further evaluation, precision and recall can be combined into an F_β score. The β can be varied to put more emphasis on either precision or recall. We follow a recent workaround mining study by weighing them equally using an F1 score (Weinzierl et al., 2022).

To determine a reasonable recall score, we need many labeled traces, but we face the same issue as before: There is no large dataset with labeled workarounds available. Creating such a dataset with domain experts would require them to evaluate a lot of traces, which would require a large time investment. In addition, since workarounds may differ strongly, it would have to be repeated for every new application (e.g., department). To solve this problem, we propose to use the discovered workaround types to generate labels for a test set.

When a workaround type is discovered, new instances of that type may be discovered using a rule specific to the type. Any trace that the expert indicates as a workaround for reasons that do not fall under an existing rule is considered a new type. For example, after taking measurements of a patient, a nurse may be required to log them as soon as possible. While a small delay can be acceptable, an expert may be able to explain that *a delay of more than an hour always means that the measurement was written down somewhere else before recording it into the system*. We can define such a rule together with the expert as soon as they indicate a trace is a workaround for a new reason.

Using such rules, we can identify all instances of the discovered workaround types. Since these instances were not included during the training of the model, they can serve as a testing set containing traces that an expert would classify as a workaround. Note that this approach will not return all workarounds, only a complete set of the workaround types that were discovered beforehand. Even though this approach will most likely not result in the true recall value of the model, it does reflect all workarounds that the model should be able to classify.

Table 1. Two example patterns for each perspective in the SWORD framework (van der Waal et al., 2022).

Detection pattern	**Explanation**
Control-flow perspective	
Occurrence of an activity	A specific activity occurs
Activity frequency out of bounds	There is a deviation in the frequency of an activity within a trace
Data perspective	
Data object with value outside boundary	The value of a data object deviates from the usual values
Change in value between events	Data values change unexpectedly between events
Resource perspective	
Activity executed by unauthorized resource	An activity is executed by a resource other than those authorized
Activities executed by a single resource	Activities within the same trace are all executed by the same resource
Time perspective	
Occurrence of activity outside of time period	An activity occurs outside of the usual time period
Delay between start of trace and activity is out of bounds	There is a deviation in the delay between the start of the trace and the time of an activity

Thus, this approach serves as a reasonable performance approximation.

We approximate precision using the same data by assuming any trace that is not marked by any rule as a workaround is normative. In truth, these traces are not evaluated, so these traces may still contain workarounds. As such, this is a lower-bounded approximation for precision. Note that the undershoot is inconsistent and may differ strongly per iteration. Therefore, it may not be usable to investigate the change in performance while training the model.

4. Case Study

To test our approach, we performed a case study with the University Medical Center Utrecht (UMCU), a Dutch academic teaching hospital with around 12,000 employees, which cares for more than 220,000 patients annually. Before we started, our study was positively evaluated according to the Medical Research Ethics Committee (MREC) NedMec policy (research protocol number 22/1055). In the UMCU, HiX[1] is used as the Hospital Information System (HIS) to store patient data across 386 tables. In collaboration with a data manager, we extracted data covering one September for the Emergency Room (ER). This department has 19 treatment rooms and treats around 18,000 patients annually. With the data, we created an event log that contained 34,929 events for 1,489 completed cases. To evaluate our findings, we interviewed a specialist in acute internal medicine who works in the department and is directly involved with many of the patients. In this section, we will elaborate on how we used this dataset for testing our method.

Table 2. The patterns scores and the score assigned by the Logistic Regression model.

Case ID	P1	P2	...	LR score
Case-0001	2.294017	1.672582	...	0.000
Case-0002	6.641274	0.643885	...	1.000
...	...	...	...	...
Case-0728	1.228111	0.384811	...	0.997
...	...	...	...	...
Case-1488	1.000074	0.899160	...	1.000
Case-1489	1.143432	0.899160	...	0.000

4.1. Constructing the Model

To construct a classification model for the ER, we started by creating a set of workaround indication features for each trace in the event log. In this study, we did this by applying the SWORD framework. Each pattern requires certain data to be available in the event log. For example, the "Event executed by a single resource" pattern requires a resource, i.e., a process participant, linked to each activity, whereas the "Event executed by an unauthorized resource" requires at least resource roles to be available. Taking such requirements into account, we could apply nine of the patterns of the SWORD framework:

- Activity frequency out of bounds
- Occurrence of a directly repeating activity
- Activities executed by multiple resources
- Activities executed by a single resource
- Occurrence of activity outside of time period
- Delay between start of trace and activity is out of bounds

[1] https://chipsoft.com/solutions

- Time between activities out of bounds
- Duration of trace out of bounds
- Delay between event and logging is out of bounds

The "Occurrence of a directly repeated activity" pattern resulted in the number of direct repeats that occurred within a trace. All remaining patterns each resulted in a Z-score that reflected the deviation from the norm per trace, effectively determining the bounds based on the data. These scores served as input for the Logistic Regression model.

4.2. Training the Model

We trained the model through Active Learning by presenting the trace with the highest expected information gain to the involved expert. In a single one-hour session, the expert evaluated 26 traces: 12 traces were considered normative, 14 contained at least one workaround. In total, we discovered seven workaround types, which can be seen in Table 3. When a new workaround type was discovered during the interview, we defined a rule to find similar instances of the type with the expert. These rules were used during the evaluation of the model for labeling test data.

4.3. Evaluating the Model

After training the model, we construct a test set by applying the expert-defined rules to the 1,463 unlabeled traces. Only if a rule showed that a workaround type occurred within the trace, it was marked as a workaround. In total, the test set contained 985 workaround traces. With this, we calculated the approximate precision and recall of the model after each training iteration, the results of which are shown in Figure 2.

The points on the left of the dotted line indicate the "warming-up" iterations. During these, the model was primarily trained on workarounds. This leads to it classifying all traces as a workaround and thus reaching a perfect recall score. Starting from iteration 5, we see a steep curve, showing quick improvement in performance, which stabilizes around 0.85 from iteration 11 onward.

The approximate precision barely changes over the training iterations. As stated earlier, this value is a lower bound on the actual precision, which may therefore be higher. Nevertheless, the minimum precision for the final model is about 0.67. This means that the corresponding F1 score for this model is 0.75.

To investigate the performance of the model on each workaround type, we determined the approximate recall per type for the final model. The results are in Table 4 with an overview of in which training iterations the workarounds occur. Since we only classify as "workaround" or "normative" and do not further distinguish between these types, we cannot determine an accompanying precision and F1 score. However, these scores can be used to compare performance between types. Note that within our context of workaround mining, recall is more important than precision; A normative trace that has been falsely identified as a workaround can quickly be dismissed by an expert, but a missed workaround may not be found at all.

5. Discussion

In this study, we combined multiple features that can indicate workarounds using Logistic Regression. To train the model with limited data availability and time involvement of a domain expert, we applied Active Learning. We evaluated this approach on real hospital data during a case study at the emergency room of the UMCU and found an approximate F1 score of 0.75 after training the model. In the remainder of this section, we discuss the implications and limitations of our work.

5.1. Implications

Our study has several implications we would like to discuss. First, while the recall scores per workaround type differ, all types are recognized with a score of at least 0.75. This shows that the model can be used even for strongly varying types of workarounds. In addition, we notice that a workaround type can be recognized with few examples in the training set. This is especially clear from workaround type 3, which occurs only in training iteration 19 but is recognized with a recall of 0.81. This is multiple rounds after the model stabilizes. Even before the expert marked this new workaround type, it is already classified correctly. This shows that the model can also classify traces that contain workarounds that have not been discovered yet. Therefore, investigating high-ranked traces that do not fit any of the known workarounds can lead to new workarounds that may be interesting for process managers.

Second, our framework can be applied with fewer requirements than the existing techniques. The patterns by Outmazgin and Soffer (2014) require boundaries to be set manually and the approach by Wijnhoven et al. (2023) requires a process model to be discovered beforehand.We can make a more quantitative comparison with Weinzierl et al. (2022). The performance of the neural network approach differs strongly with F1 scores ranging from 0.33 to 0.83. These scores vary strongly depending on the exact dataset used. Our score of 0.75 is slightly lower than their highest score, but much higher than the lowest. Since we used different data, these scores are

Table 3. The workaround types discovered at the ER, their explanation, and the rule to discover similar instances. Note that rules have a little slack time (10 - 15 minutes) to account for the order of activity registrations. For example, if an activity can only occur after a patient was seen by a nurse, the activity may be performed first and the checkmark placed directly afterward.

	Workaround type	Reasoning	Rule
1	Activities after discharge	Activities cannot be performed on a patient after they have left the ER. If this is observed in a trace, registration was late.	If an activity occurs after a patient has been discharged, there is a workaround.
2	Orders before the patient was seen by a physician and after initial intake	Orders can be placed in two ways: 1) a physician inspects the patient and places orders, or 2) the patient is expected and tests were ordered before their arrival.	If an order is placed more than 10 min before "Seen by physician", there is a workaround. Exception: Orders placed within 10 min of triage are normative.
3	Extra tests performed before a patient was seen by a physician	Nurses generally only order standard tests. A physician may determine that "extra tests" are required. These should naturally only occur after they saw the patient.	If the "extra test" activity occurs more than 10 min before "Seen by physician", there is a workaround.
4	Late triage registration	Triage (determining the urgency of care) always occurs at the start of a treatment process. It should also be registered quickly.	If the registration of the triage is performed after more than 15 min after the first activity by a nurse, there is a workaround.
5	Activities were executed before the patient was seen	Similar to the previous type, nurse activities, such as taking measurements, should only occur after they saw the patient	If triage, measurement, or order is registered more than 10 min before "Seen by nurse", there is a workaround.
6	Discharge without two pain scores	Any patient in the ER is asked twice how much pain they have to see if any needs to be addressed and to see if treatment helped.	If a patient was discharged with fewer than two registered pain scores, there is a workaround.
7	Immediate release after an x-ray	After an x-ray is performed, there is some time required to interpret and explain the results before a patient can be released.	If a patient was released less than 20 min after an x-ray was taken, there is a workaround.

not sufficient to conclude that either approach performs better in general, but we can see our performance is in the same range. In addition, it should be noted that the neural network approach has been tested on partially synthetic data, so it is unclear how it performs on real data. One advantage of our model is that it requires relatively little data. Where our model stabilizes after 11 instances, the neural network is trained on 7,000 to 16,000 instances. Even if our approach does not match the optimal performance, the low data requirements make it easier to apply in a practical scenario.

Finally, while any feature-based model can be used to indicate traces of interest, domain experts are required for actual classification. To limit the time and effort requirements of these experts, we optimize training the model by only querying the most valuable traces using Active Learning. With nine features as input, the performance of the model still stabilizes after only 10 to 15 items, which we reached well within a single one-hour session with a domain expert. This shows that training a model with Active Learning to recognize workarounds requires little time and data, especially when compared to qualitative studies that spent between 20 and 30 hours on interviews and observations (Beerepoot et al., 2019; Beerepoot & van de Weerd, 2018). This makes it easier for process managers to apply our research in practice.

5.2. Limitations

As with most studies, there are limitations to this work, which we will discuss in this section. First, we can only approximate precision and recall in this study. We cannot calculate a true score for either because this would require labeling all workaround traces in the log. Recall varies strongly over time and follows a curve that shows a learning effect. However, precision varies very little. It is around 0.67 during all training iterations. We see two possible explanations for this behavior. 1) The model learns to recognize workarounds, but not normative traces, or 2) The test set, which consists of 67% workarounds, is not suitable to approximate precision. The second option would also explain why the F1 score during the warming-up phase is higher than after the model is trained. As to the impact of this on our research, we stated earlier that the approximate precision does provide a lower bound. This means the F1 score can still

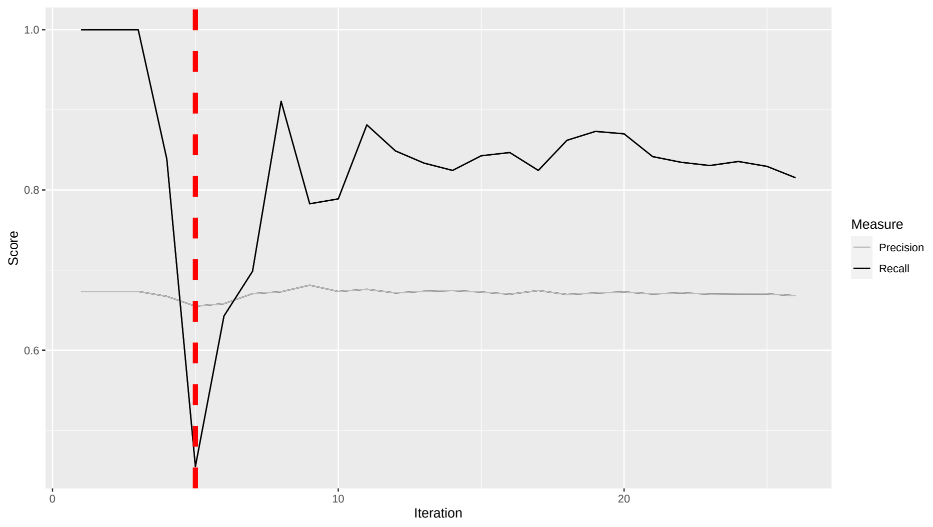

Figure 2. The approximate precision and recall of the model. The x-axis shows how many traces the model is trained on, the y-axis shows the approximate score for precision and recall. The focus is on the right side of the dotted line. The training iterations before this are part of the "warming-up" phase of the model.

Table 4. The approximate recall of the final model per workaround type.

	Workaround Type	Occurs in Training Iteration	Approx. Recall
1	Activities after discharge	1, 7, 10	0.86
2	Orders before the patient was seen by a physician and after initial intake	2, 5, 8, 15, 17, 19	0.76
3	Extra tests performed before a patient was seen by a physician	19	0.81
4	Late triage registration	5, 11, 15, 17	0.75
5	Activities were executed before the patient was seen	5, 8, 11, 16, 17	0.76
6	Discharge without two pain scores	5, 24	0.81
7	Immediate release after an x-ray	3, 6	0.82

be used to give a lower bound for our results.

The second limitation is related to the use of event logs for the discovery of workarounds. We discover seven workaround types in this study but based on previous research in the same department (van der Waal et al., 2023), we are aware of more workarounds that occur there. Some of these workarounds could likely be discovered with a second interview, but other workarounds cannot be discovered at all through event logs. For example, a nurse that calls the pharmacy for current information concerning a medication order instead of looking it up in the HIS can never be detected if these activities are not logged. While we show that we can discover new workarounds using our approach, there is no guarantee we find all workarounds in the data.

Thirdly, the workarounds we can find with our approach depend heavily on the knowledge of the domain expert. We interviewed a physician who was aware of many of the workarounds that other workers, such as nurses, employ. If an expert is less familiar with their day-to-day tasks, evaluating traces that primarily focus on their process will be difficult to classify. To put the results of our approach to use, expert involvement will always be of paramount importance conform Bezemer et al. (2019). Even though our approach minimizes experts' involvement, it still relies on them for evaluation, so the involved experts should be carefully selected.

Finally, we only evaluate a single ward in this study. Previous research shows that different contexts may have strong effects on performance (Weinzierl et al., 2022), which makes generalizing the results of detection models problematic. During this study, we also investigated data from the Critical Care Unit (CCU). Because the care is by definition critical in the department, there is a strong sense of urgency to follow the official procedures, which leads to few workarounds in this department.

6. Conclusion

With this study, we extend the field of quantitative workaround mining by creating a Logistic Regression model that combines multiple features that are known to be useful when discovering workarounds. To train such a model, we required labeled training data, which is not readily available. We make the most use of the experts' limited time by training the model through Active Learning. In this way, domain experts only need to evaluate the traces that provide the most information to the model. In a case study, we approximate the performance of the model by creating a test set based on the workaround types we discovered during a one-hour interview with a domain expert. Using this approximation, we find an F1 score of 0.75 after training the model. To conclude, we propose multiple directions for future research: the evaluation can be broadened, expert involvement can be improved, and there are alternatives to several choices for our approach.

First, as we state in the discussion, we only test this approach in one case. The evaluation can be improved by investigating different departments or even different domains. We expect that the performance can change depending on how common workarounds are and what types of workarounds occur most often. While we initially tried this at the CCU, the related evaluation was limited by the few workarounds that are applied here. Thus, it is important to note that this should be tested in a context where we can expect workarounds to occur.

Second, during the interview, the domain expert did not feel like the presented rank and standardized pattern scores were useful to determine what happened in a trace. Instead, they felt more comfortable looking through a full trace without focusing on any part indicated by the patterns. In future work, it may be more useful to give more specific guidance. For example, if an activity occurs relatively late in a trace, this specific deviation could be highlighted instead of using abstract scores.

Finally, we suggest six alternatives to the choices we made: (1) Precision and recall may be approximated by querying a small number of random traces to the expert. We decided against this approach in this work because we lacked information concerning the ratio of workaround traces compared to normative traces. If this ratio is strongly skewed to either side, an approximation with too few traces would not be accurate. This made it difficult to determine how many traces should be labeled by the expert (Powers, 2011). In addition, it was unclear how much time an expert would require to label a trace. Since their time is limited, we opted for our rule-based evaluation. (2) To score the performance of our model, we used F1 scores to strike a balance between precision and recall. However, there are many alternatives, such as F scores (Verbeek, 2020) or Cohen's Kappa (Warrens, 2015). We decided to use F1 scores to be able to compare our approach with the existing workaround detection technique by Weinzierl et al. (2022). (3) Based on the discovered workaround types, we define rules with the expert to create a test set. The domain experts were confident about the rules they posed, but it may be interesting to look into rule-mining (Alman et al., 2020) instead to automate this process and reduce the expert's time investment even further. (4) For reasons we outline in this paper, we decided to create our model based on Logistic Regression. More complex models, such as Support Vector Machines or Decision Trees can outperform simple models, but they may require more labeled data (Sordo & Zeng, 2005). (5) Both previous suggestions require more data than our current approach. In this paper, we optimize data selection through Active Learning. Alternatively, we could construct a dataset through, for example, crowdsourcing with medical students. While the individual quality of the students would be lower, the group may be able to provide quality labels in a larger quantity than possible for "expensive" physicians. (6) To create our model, we have only investigated supervised techniques. However, unsupervised methods may be an alternative option to circumvent the data requirements. Methods such as trace clustering (Song et al., 2009) may be used to train models without labeled data or it could be used as a starting point to further optimize evaluating traces with domain experts.

6.0.1. Acknowledgements We would like to thank M.C.H. de Groot, PhD for his help during data extraction, as well as B. Vrijsen, MD and G.H.J.M. Ellenbroek MD, PhD for their time to train the model and discuss our findings.

This publication is part of the WorkAround Mining (WAM!) project with project number 18490 which is (partly) financed by the Dutch Research Council (NWO).

References

Alman, A., Di Ciccio, C., Haas, D., Maggi, F. M., & Nolte, A. (2020). Rule mining with RuM. *International Conference on Process Mining (ICPM)*, 121–128.

Alter, S. (2014). Theory of Workarounds. *Communications of the Association for Information Systems (CAIS)*, *34*.

Azad, B., & King, N. (2008). Enacting computer workaround practices within a medication dispensing system. *European Journal of Information Systems (EJIS)*, *17*(3), 264–278.

Beerepoot, I., Lu, X., van de Weerd, I., & Reijers, H. A. (2021). Seeing the Signs of Workarounds: A Mixed-Methods Approach to the Detection of Nurses' Process Deviations. *Proceedings of the Annual Hawaii International Conference on System Sciences (HICSS)*.

Beerepoot, I., Ouali, A., van de Weerd, I., & Reijers, H. A. (2019). Working Around Health Information Systems: To Accept Or Not To Accept? *Twenty-Seventh European Conference on Information Systems*.

Beerepoot, I., & van de Weerd, I. (2018). Prevent, Redesign, Adopt or Ignore: Improving Healthcare using Knowledge of Workarounds. *Proceedings of the 26th European Conference on Information Systems (ECIS)*.

Bezemer, T., de Groot, M. C. H., Blasse, E., ten Berg, M. J., Kappen, T. H., Bredenoord, A. L., van Solinge, W. W., Hoefer, I. E., & Haitjema, S. (2019). A Human(e) Factor in Clinical Decision Support Systems. *Journal of Medical Internet Research (JMIR)*, *21*(3), e11732.

Ejnefjäll, T., & Ågerfalk, P. J. (2019). Conceptualizing Workarounds: Meanings and Manifestations in Information Systems Research. *Communications of the Association for Information Systems (CAIS)*, 340–363.

Kleinbaum, D. G., & Klein, M. (2002). *Logistic Regression: A Self-Learning Text* (Third). Springer.

Lewis, D. D. (1998). Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval. In *Machine learning: ECML-98* (pp. 4–15). Springer Berlin Heidelberg.

Lewis, D. D., & Catlett, J. (1994). Heterogeneous Uncertainty Sampling for Supervised Learning. In *Machine learning: Proceedings of the eleventh international conference* (pp. 148–156). Elsevier.

Outmazgin, N., & Soffer, P. (2014). A process mining-based analysis of business process work-arounds. *Software & Systems Modeling (SoSyM)*, *15*(2), 309–323.

Powers, D. M. W. (2011). Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. *International Journal of Machine Learning Technology*, *2*, 2229–3981.

Röder, N., Wiesche, M., & Schermann, M. (2014). A Situational Perspective on Workarounds in IT-Enabled Business Processes: A Multiple Case Study. *Proceedings of the 22nd European Conference on Information Systems (ECIS)*.

Schein, A. I., & Ungar, L. H. (2007). Active learning for logistic regression: An evaluation. *Machine Learning*, *68*(3), 235–265.

Settles, B. (2009). *Active learning literature survey* (Computer Sciences Technical Report No. 1648). University of Wisconsin–Madison.

Song, M., Günther, C. W., & van der Aalst, W. M. P. (2009). Trace Clustering in Process Mining. *Business Process Management Workshops*, 109–120.

Sordo, M., & Zeng, Q. (2005). On Sample Size and Classification Accuracy: A Performance Comparison. *Biological and Medical Data Analysis (ISBMDA 2005)*, 193–201.

van der Waal, W., Beerepoot, I., van de Weerd, I., & Reijers, H. A. (2022). The SWORD is Mightier Than the Interview: A Framework for Semi-automatic WORkaround Detection. *Business Process Management (BPM)*, 91–106.

van der Waal, W., van de Weerd, I., Beerepoot, I., Kappen, T., Haitjema, S., & Reijers, H. A. (2023). Putting the SWORD to the Test: Finding Workarounds with Process Mining [Manuscript submitted for publication].

van de Weerd, I., Vollers, P., Beerepoot, I., & Fantinato, M. (2019). Workarounds in retail work systems: Prevent, redesign, adopt or ignore? *Proceedings of the 27th European Conference on Information Systems (ECIS)*.

Verbeek, H. M. W. (2020). Process discovery contest 2020.

Warrens, M. J. (2015). Five ways to look at Cohen's kappa. *Journal of Psychology & Psychotherapy*, *5*(4), 1.

Weinzierl, S., Wolf, V., Pauli, T., Beverungen, D., & Matzner, M. (2022). Detecting temporal workarounds in business processes – A deep-learning-based method for analysing event log data. *Journal of Business Analytics (JBA)*, *5*(1), 76–100.

Wijnhoven, F., Hoffmann, P., Bemthuis, R., & Boksebeld, J. (2023). Using process mining for workarounds analysis in context: Learning from a small and medium-sized company case. *International Journal of Information Management Data Insights (IJIM Data Insights)*, *3*(1), 100163.

Wolf, V., & Beverungen, D. (2019). Conceptualizing the impact of workarounds: An organizational routines perspective. *Proceedings of the 27th European Conference on Information Systems (ECIS)*.