Interactive Rule Refinement for Fraud Detection

Tova Milo
Tel Aviv University
milo@post.tau.ac.il

Slava Novgorodov
Tel Aviv University
slavanov@post.tau.ac.il

Wang-Chiew Tan
Recruit Institute of Technology
wangchiew@recruit.ai

ABSTRACT

Credit card frauds are unauthorized transactions that are made or attempted by a person or an organization that is not authorized by the card holders. Fraud with general-purpose cards (credit, debit cards etc.) is a billion dollar industry and companies are therefore investing significant efforts in identifying and preventing them.

It is typical to deploy mining and machine learning-based techniques to derive rules. However, such rules may not always capture the semantic reasons underlying the frauds that occur. For this reason, credit card companies often employ domain experts to manually specify rules that exploit general or domain knowledge for improving the detection process. Over time, however, as new (fraudulent and legitimate) transactions arrive, these rules need to be updated and refined to capture the evolving (fraud and legitimate) activity patterns. The goal of the RUDOLF system described in this paper is to guide and assist domain experts in this challenging task. RUDOLF automatically determines the "best" adaptation to existing rules to capture all fraudulent transactions and, respectively, omit all legitimate transactions. The proposed modifications can then be further refined by users and the process can be repeated until they are satisfied with the resulting rules. We show that the problem of identifying the best candidate adaptation is NP-hard in general and present PTIME heuristic algorithms for determining the set of rules to adapt. We have implemented our algorithms in RUDOLF and show, through experiments on real-life datasets, the effectiveness and efficiency of our solution.

1 INTRODUCTION

A *credit card fraud* is an unauthorized transaction made or attempted by an individual or organization who is not authorized by the card holder to use a credit card to perform the electronic payment. Fraud with general-purpose cards (credit, debit cards etc.) is a billion dollar industry. In fact, several independent news articles and studies that were carried out (e.g., [1, 2]) corroborate that there is a consistent, fast-growing, and upward trend on the total global payment-card frauds. Detecting and deterring credit card frauds are therefore of extreme importance to credit card companies. A core part of operations behind every credit card company is to (automatically) detect fraudulent transactions among the new transactions (e.g., [3]) that are received everyday.

To this end, models based on data mining and machine-learning techniques (e.g. [4–6]) have been used. A typical approach is to score each transaction where transactions whose scores are above a threshold are classified as fraudulent. However, the models and scoring system do not always have high precision and recall. Fraudulent transactions may be missed by the models and, likewise, legitimate transactions may be wrongly identified as fraudulent. The derived threshold also do not provide a semantic explanation of the underlying causes of the frauds. It is for this reason that credit card companies typically rely on *rules* that are carefully specified by domain experts in addition to models for automatically determining fraudulent transactions.

Intuitively, a rule describes a set of transactions in the database and the goal is to arrive at a set of rules that, together with the automatically derived scores, captures precisely the fraudulent transactions. The use of rules written by users has the advantage that it allows employing general or domain knowledge to handle rare special cases.

Writing rules to capture precisely fraudulent transactions is a challenging task that is exacerbated over time as the types of fraudulent transactions evolve or as new knowledge is learnt. Typically, a set of rules that were curated by users already exists and the rules work well for capturing fraudulent transactions up to a certain day. However, these rules need to be adapted over time to capture new types of frauds that may occur. For example, there may be new reported fraudulent transactions coming from a certain type of store at a certain time that did not occur before and hence, not caught by existing rules. Analogously, there may be some transactions that were identified by the existing rules as fraudulent but later verified by the card holders to be legitimate. Hence, the rules have to be adapted or augmented over time to capture all (but only) fraudulent transactions. In this paper, we present RUDOLF, a system whose goal is to assist users to define and refine rules for fraud detection.

Note that our goal resembles in part that of previous works on query/rule refinement, which attempt to automatically identify minimal changes to the query/rule in order to insert or remove certain items from the query result (e.g., [7]). However, a key difference here is that such minimal modifications often do not capture the actual "ground truth", namely the nature of ongoing attack, which may not yet be fully reflected in the data. By interacting with users to fine-tune rules, important domain knowledge can be effectively imported into the rules to detect the pattern of frauds often even before they are manifested in the transactions themselves.

Our goal also resembles previous work on discovering or learning decision rules from streams with concept drifts (e.g. [8, 9]). Like them, RUDOLF strives to discover or adapt rules as new transactions arrive. However, all previous work considered only domains over numerical values and hence do not immediately apply to our setting, which involves both categorical and numerical values. Furthermore, RUDOLF makes crucial use of a hierarchy of higher-level concepts over the domains (numerical or categorical) in the specification of rules. In addition, RUDOLF collaborates with the domain expert to improve upon the quality of the rules used for capturing only fraudulent transactions. The interplay between the domain experts and the use of higher-level concepts, whenever possible, enables the derivation of rules which can be used to explain the true nature ongoing frauds. Our experiments indicate that such interactions can be effective in deriving rules with good prediction quality.

An overview example The top of Figure 1 shows a simplified set of rules that is currently used to capture fraudulent transactions up to yesterday. Intuitively, the first two rules capture a suspicion of two attacks on an online store taking place at the first and

last few minutes of 6pm, charging amounts over $110. The last rule captures a fraud pattern at Gas station A where false charges of amounts over $40 are made soon after the closing time at 9pm. In practice each rule also includes some threshold condition (not shown) on the score (i.e., the degree of confidence that the transaction is fraudulent) for each transaction, as well as additional conditions on the user/settings/etc. The scores and the additional conditions are omitted so that we can focus our discussions on the semantic aspect of the rules shown in Figure 1.

Figure 2 shows an example of a relation which contains a number of transactions made today. The transaction tuples are ordered by the time of the transaction. In the figure, some transactions that were reported as fraudulent are labeled as "FRAUD". Similarly, transactions that are reported to be legitimate may be correspondingly labeled "LEGITIMATE" (not shown in the figure). Transactions may also be unlabeled. The current set of rules captures only the shaded tuples shown in the transaction relation. Clearly, none of the new fraudulent transactions are captured by the existing rules whereas some unlabeled transactions are captured.

RUDOLF first attempts to capture the fraudulent transactions by generalizing the rules, semantically according to a given ontology whenever possible, before it specializes the rules to avoid unnecessarily capturing legitimate transactions. However, the changes proposed by RUDOLF may not correspond to the best or correct changes. The domain experts can view/accept/reject/modify the suggestions provided by RUDOLF, arriving for instance at the
1) Time $\in$ [18:05,18:05] $\wedge$ Amt $\geq$ 100 $\wedge$ Type = Onl., no CCV.
2) Time $\in$ [18:55,19:15] $\wedge$ Amt $\geq$ 100 $\wedge$ Type = Onl., no CCV.
3) Time $\in$ [20:45,21:15] $\wedge$ Amt $\geq$ 40 $\wedge$ Location $\leq$ Gas Station $\wedge$ Type $\leq$ Offline.

Intuitively, the first two rules above flag online transactions without CCV, charging amounts over $100 in the respective time intervals as fraudulent transactions. The third rule flags offline transactions at the gas stations around closing time of amounts over $40 as fraudulent transactions. Observe that the condition "Location $\leq$ 'Gas Station' is a semantic generalization of Gas Stations A and B, which are defined in an ontology to be contained within the category "Gas Station". Similarly, "Type $\leq$ Offline" reflects the semantic category (shown at the bottom of Figure 1) which contains offline transactions with and without PIN.

Contributions This paper makes the following contributions.

(1) We formulate and present a novel interactive framework for determining the "best" way to adapt and augment rules so that fraudulent transactions are captured and, at the same time, legitimate transactions are avoided.
(2) We establish that the rule refinement problem is NP-hard even under special circumstances: (1) determine the best way to generalize rules to capture new fraudulent transactions when there are no new legitimate transactions, and (2) determine the best way to specialize existing rules to avoid capturing new legitimate transactions when there are no new fraudulent transactions.
(3) In light of these hardness results, we develop a heuristic algorithm which is able to interactively adapt rules with domain experts until a desired set of rules is obtained. At each step, the algorithm makes a proposal of the best changes to a rule, and the domain expert can further refine the proposed changes or seek suggestions for other possible changes. Our algorithm represents a departure from prior algorithms on discovering or learning decision rules from streams with concept drifts in that it handles categorical values in addition to numerical values, adapt rules with semantic concepts from available ontologies, and interacts with domain experts.
(4) We have implemented our solution in the RUDOLF prototype system and applied it on real data, demonstrating the effectiveness and efficiency of our approach. We performed experimental evaluations on a real-life dataset of credit card transactions. We show that by interacting with users (even ones with only little knowledge specific to the domain of the datasets), our algorithms consistently outperform alternative baseline algorithms, yielding more effective rules in shorter time.

While most of our exposition on the features of RUDOLF is based on credit card frauds, we emphasize that RUDOLF is a general-purpose system that can be used to interact with users to refine rules. For example, for preventing network attacks, for refining rules for spam detection or for intrusion detection [10].

A first prototype of the system was demonstrated at VLDB'16 [11]. The short paper accompanying the demonstration gives only a brief overview of the system architecture whereas the present work provides a comprehensive description of the underlying model and algorithms.

The paper is organized as follows. The next two sections define the model and problem statement behind RUDOLF (Section 2 and, respectively, Section 3). The algorithm behind RUDOLF is described in Section 4. We then present our experimental results (Section 5) and related work (Section 6), before we present our conclusions (Section 7).

2 PRELIMINARIES

Transaction relation A *transaction relation* is a set of tuples (or *transactions*). The transaction relation is appended with more transactions over time. We assume that the domain of every attribute A has a partial order, which is reflexive, antisymmetric, and transitive, with a greatest element $\top_A$ and least element $\bot_A$. W.l.o.g. we also assume that $\bot_A$ does not appear in any of the tuples[1]. For brevity, when an attribute name is clear from the context we will omit it and simply use the notations $\top$ and $\bot$. Attributes that are associated with a partial order but not a total order are called *categorical attributes*. The elements in such partial order are sometimes referred to as *concepts*.

A transaction may be flagged as *fraudulent* which means that the transaction was carried out illegally or conversely, a transaction may be flagged as *legitimate*. Unmarked transactions are called *unlabeled* transactions. The labeling is assumed to correspond to the (known part of the) *ground truth*. In addition, each transaction has a `score` between 0 and 1, that is computed automatically using machine learning techniques, and depicts the estimated probability of each transaction to be fraud. The score may or may not agree with the ground truth and this discrepancy is precisely the reason why rules are employed to refine the fraud detection.

Example 2.1. Part of a transaction relation I with schema `T(time,amount,type,location,...)` is shown in Figure 2. Each tuple records, among others, the time, amount, type of transaction, and location where the purchase was made through some credit card. The scores of the transactions, as computed by a machine learning module, are omitted from the figure. The last column annotates the type of transactions. The part of instance I that is shown contains only fraudulent and unlabeled transactions.

[1] If this is not the case, add a new special element to the domain and set it smaller, in the partial order, than all other elements.

Existing fraud rules Φ from the previous day:
1) Time ∈ [18:00, 18:05] ∧ Amt ≥ 110
2) Time ∈ [18:55, 19:00] ∧ Amt ≥ 110
3) Time ∈ [21:00, 21:15] ∧ Amt ≥ 40 ∧ Location≤'Gas Station A'

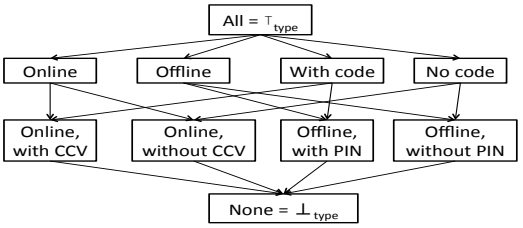


Figure 1: Top: An existing set of rules. Bottom: Partial Order for `type` values.

Time	Amount	Transaction Type	Location	
18:02	107	Online, no CCV	Online Store	FRAUD
18:03	106	Online, no CCV	Online Store	FRAUD
18:04	112	Online, with CCV	Online Store	
19:08	114	Online, no CCV	Online Store	FRAUD
19:10	117	Online, with CCV	Online Store	
20:53	46	Offline, without PIN	GAS Station B	FRAUD
20:54	48	Offline, without PIN	GAS Station B	FRAUD
20:55	44	Offline, without PIN	GAS Station B	FRAUD
20:58	47	Offline, with PIN	Supermarket	
21:01	49	Offline, with PIN	GAS Station A	
:	:	:	:	:

Figure 2: A transaction relation containing new transactions.

The type `time` is discretized to minutes and is associated with a date (not shown). It thus has a partial order (in fact, a total order), given by the ≤ relation. The type `amount` also has a total order with least element 0 and greatest element ∞. The attribute `type` is a categorical attribute and its partial order is given by the hierarchy shown at the bottom of Figure 1. Some examples of concepts in the hierarchy are "Online" or "Offline, without PIN". The type `location` is also a categorical attribute and its partial order (not shown) is given by, say, the geographical containment relationship. In particular, "Gas Station A" and "Gas Station B" are both concepts that are children of the concept "Gas Station".

Rules For simplicity and efficiency of execution, we assume rules are typically written over a single relation, which is a universal transaction relation that includes all the necessary attributes (possibly aggregated or derived from many other database relations) for fraud detection. Hence, it is not necessary to consider explicit joins over different relations in the rule language.

To highlight the key principles of our approach we consider here a fairly simple rule language that captures a disjunction of conjunctions. For simplicity, our rule language allows only one condition over each attribute. Multiple disjunctive conditions over the same attribute can be expressed in multiple rules. Other extensions to the rule language are possible but will not be considered here. Note that the rule language that we consider, albeit simple, forms the core of common rule languages used by actual systems.

A *rule* is a conjunction of one or more conditions over the attributes of the transaction relation. More precisely, a rule is of the form $\alpha_1 \wedge \ldots \wedge \alpha_n$ where n is the arity of the transaction relation, α_i is a condition is of the form 'A_i *op* s' or '$A_i \in [s, e]$', A_i is the ith attribute of the transaction relation, $op \in \{=, <, >, \leq, \geq\}$, and s and e are constants.

More formally, if φ is a rule that is specified over a transaction relation I, then $\varphi(I)$ denotes the set of all tuples in I that satisfy φ. We say that $\varphi(I)$ are the transactions in I that are *captured* by φ. If Φ denotes a set of rules over I, then $\Phi(I) = \bigcup_{\varphi \in \Phi} \varphi(I)$. In other words, $\Phi(I)$ denotes the union of results of evaluating every rules in Φ over I. Observe that $\Phi(I) \subseteq I$ since every rule selects a subset of transactions from I. For readability, in our examples we show only the non trivial conditions on attributes, namely omit conditions of the form $A_i \leq \top$.

Note that, for simplicity, each rule includes only one condition over each attribute, but multiple disjunctive conditions over the same attribute can be expressed using multiple rules.

Example 2.2. The top of Figure 1 illustrates a set Φ of three simplified rules currently used by the example credit card company to detect fraudulent transactions. The first rule captures all transactions made between 6pm to 6:05pm where the amount involved is at least $110. As previously mentioned, in practice each rule also includes some threshold conditions (not shown here) on the score for each transaction, as well as additional conditions on the user/settings/etc. For simplicity we omit the score thresholds and the additional conditions and focus in the sequel on the simplified rules in this example.

For the new transaction relation shown in Figure 2, this rule captures the 3rd tuple (which is an unlabeled transaction). The 2nd rule captures no tuples, and the 3rd rule captures the 10th unlabeled tuple. Hence, the existing rules Φ do not capture any of the fraudulent transactions on the current day.

As we shall demonstrate, the rule language that we consider here, even though simple, is able to succinctly capture the fraudulent transactions (and avoid legitimate tuples) in our experiments with a real dataset. Other domains rules, e.g. access control for network traffic or spam detection rules can also be expressed in our language.

Cost and Benefit of modifications A *modification* to a rule is a change in a condition of an attribute in the rule. One may also *copy* an existing rule before modifying the copy, add rule or remove an existing rule. As we will elaborate in subsequent sections, our cost model assumes there is a cost associated with every operation/modification made to a condition in the rule.

To compare between possible modifications to rules and determine which modifications are better, we need to know not only the cost of each modification but also the "benefit" it entails. Intuitively, the gain from the modifications can be measured in three ways: (1) the increase in the number of fraudulent transactions that are captured by the modified rule, (2) the the decrease in the numbers of legitimate transactions that are captured by the modified rule, and (3) the decrease in the numbers of captured unlabeled transactions. The assumption underlying (3) is that unlabeled transactions are typically assumed to be correct until explicitly reported/tagged otherwise and the more specific the rules are to fraudulent (and only) fraudulent transactions, the more precise they are in embodying possible domain-specific knowledge about the fraudulent transactions.

Observe that if our modifications are ideal, then after the modifications, there are more fraudulent transactions captured, and less

legitimate and unlabeled transactions caught. Subsequently, the overall update cost is defined as the cost of modifications minus their benefit. We give the formal definition in the next section.

3 PROBLEM STATEMENT

As described, the goal of RUDOLF is to identify minimal modifications to existing rules that would ideally capture all fraudulent transactions, omit all legitimate transactions, and at the same time, minimize the inclusion of unlabeled transactions. The modifications suggested by RUDOLF serve as a starting point for the domain expert who can either accept the proposed modifications or interactively refine them with the help of RUDOLF . Formally, the problem is stated below.

Definition 3.1 (General Rule Modification Problem). Let Φ be a set of rules on an existing transaction relation I. Let I' denote a new set of transactions over the same schema. Let $F, L \subseteq I$ (resp. $F', L' \subseteq I'$) be two disjoint sets of *fraudulent* and *legitimate* transactions in I (resp. I'). Let $R = I - (F \cup L)$ (resp. $R' = I' - (F' \cup L')$) be the remaining *unlabeled* transactions in I (resp. I').

The GENERAL RULE MODIFICATION PROBLEM is to compute a set M of modifications to Φ to obtain Φ' so that $\text{cost}(M) - (\alpha * \Delta F + \beta * \Delta L + \gamma * \Delta R)$ is minimized, where $\alpha, \beta, \gamma \geq 0$, and

- $\Delta F = |(F \cup F') \cap \Phi'(I \cup I')| - |(F \cup F') \cap \Phi(I \cup I')|$,
- $\Delta L = |(L \cup L') \cap \Phi(I \cup I')| - |(L \cup L') \cap \Phi(I \cup I')|$, and
- $\Delta R = |(R \cup R') \cap \Phi(I \cup I')| - |(R \cup R') \cap \Phi(I \cup I')|$.

The term $(\alpha * \Delta F + \beta * \Delta L + \gamma * \Delta R)$ represents the benefit of applying the given modifications to the rules. In our rule language, if a fraudulent transaction is not captured by a set of rules, then at least some condition of a rule needs to be generalized to select that fraudulent transaction. At the same time, making a condition more general may capture also legitimate or unlabeled transactions. Conversely, if a legitimate transaction is captured by a set of rules, then at least some condition of a rule needs to be made more restrictive so that the legitimate transaction is excluded. Such modifications carry the risk of omitting some fraudulent transactions that should be captured. The coefficients α, β and γ are non-negative and are typically provided by the user to tune the relative importance of each category (resp. correctly capturing the fraudulent transactions, avoiding misclassifying legitimate transactions, and excluding unlabeled transactions) in the calculation of benefit. The overall goal is to identify the modifications having the best cost-benefit balance.

Observe that if α, β and γ are set to large numbers (e.g. greater than the maximal update cost) then the most beneficial modifications are those leading to a "perfect" set of rules, namely one that will (1) capture all fraudulent transactions, (2) exclude all legitimate transactions, and at the same time, (3) does not capture any unlabeled transactions.

4 THE GENERAL RULE MODIFICATION ALGORITHM

The rule modification algorithm (outlined below) first interactively refines the rules to capture fraudulent transactions. The expert can stop the refinement whenever she is satisfied with the rules and believes that the omission of the remaining fraudulent transactions is tolerable. The resulting set of rules may capture some (existing) legitimate tuples. Hence, in the second step, the algorithm continues to interactively refine the rules to avoid the legitimate transactions. Here again the user may stop the algorithm when she believes that the inclusion of the remaining legitimate transactions is acceptable. However, the rules that result after this step may no longer capture some fraudulent transactions that were previously captured. The domain expert can either repeat the process described above to remedy this, or choose to end the rule refinement process at this point. In the latter case, the domain expert has a choice to leave the result as-is or allow the algorithm to create transaction-specific rules to capture each of the remaining transactions.

(1) Generalize rules to capture fraudulent transactions. See Algo. 1, section 4.1.
(2) Specialize rules to avoid legitimate transactions. See Algo. 2, section 4.2.
(3) Exit if the domain expert is satisfied. Otherwise, repeat the steps above.

Observe that it is essential for the generalization algorithm (Algo. 1) to be applied before the specialization algorithm (Algo. 2) as one can always add rules to capture specific fraudulent transactions without accidentally capturing legitimate transactions. On the other hand, one cannot always add/modify rules to avoid specific legitimate transactions without accidentally excluding fraudulent transactions with our current rule language.

As we shall show in the next sections, finding an optimal set of changes to the rules is computationally expensive in either case. For this reason, instead of computing an optimal set of modifications to the rules to generalize or special rules to capture fraudulent and, respectively, avoid legitimate tuples, we develop heuristic algorithms to identify the best update candidates in each of the cases.

4.1 Rule Generalization Algorithm

We first consider how rules can be generalized to capture fraudulent transactions when the set I' of new transactions contains only fraudulent transactions. The goal here is to adapt the existing set of rules Φ to capture all fraudulent transactions. We call this problem the RULE GENERALIZATION PROBLEM

THEOREM 4.1. *The* RULE GENERALIZATION PROBLEM *is NP-hard even if Φ is perfect for I, (namely, $F \subseteq \Phi(I)$ and $L \cap \Phi(I) = R \cap \Phi(I) = \emptyset$). The problem is NP-hard even when I contains only unlabeled transactions and I' consist of only one fraudulent transaction.*

PROOF. We prove the two claim simultaneously by reduction from the minimum hitting set problem which is known to be NP-hard. We recall of the Minimum Hitting Set Problem below.

Definition 4.2 (Minimum Hitting Set). Consider the pair (U, S) where U is a universe of elements and S is a set of subsets of U. A set $H \subseteq U$ is a hitting set of S if H hits every set in S. In other words, $H \cap S' \neq \emptyset$ for every $S' \in S$. A minimum hitting set H is a minimum cardinality hitting set s.t. $\forall e \in H$, we have $H \setminus \{e\}$ is not a hitting set of S.

We assume that each rule modification is associated with a unit cost. Given an instance of the hitting set problem, we construct an instance of the RULE GENERALIZATION PROBLEM were there no fraudulent or legitimate transactions in I and I' consist of only one fraudulent transaction, as follows.

The transaction relation has $|U|$ columns, one for each element in U. The transaction relation I has a characteristic (unlabeled) tuple for every set $s \in S$. That is, for every set s in S, we construct a characteristic tuple of s in I by placing a 0 in position i if $x_i \in s$ and 1 otherwise. Hence there are $|S|$ tuples in the transaction

relation I and these are unlabeled tuples that we would like to minimize capturing with the rules. There are no existing fraudulent or legitimate tuples in I. The set Φ is initially empty. Hence, $\Phi(I)$ does not capture any transaction (and thus, by definition, is "perfect" for I). The instance I' consists of a single transaction (1,1,1,...,1), which is the new fraudulent transaction that we wish to capture.

As an example, consider the following hitting set where $U = \{A_1, A_2, A_3, A_4, A_5\}$ and $S = \{s_1, s_2, s_3\}$, where $s_1 = \{A_1, A_2, A_3\}$, $s_2 = \{A_2, A_3, A_4, A_5\}$, and $s_3 = \{A_4, A_5\}$. The transaction relation $I \cup I'$ (where I' is highlighted in gray) is shown below. The last column annotates the type of tuple (e.g., the last tuple is a fraudulent transaction).

A_1	A_2	A_3	A_4	A_5	
0	0	0	1	1	
1	0	0	0	0	
1	1	1	0	0	
1	1	1	1	1	F'

It is straightforward to verify that the construction of the instance of the Rule Modification Problem can be achieved in polynomial time in the size of the input of the Hitting Set Problem. We now show that a solution for the Minimum Hitting Set Problem implies a solution for the Rule Modification Problem and vice versa.

Let H be the minimum hitting set. For every element x_i in H, we add a condition $a_i = 1$ to the same rule. (The condition can also be $a_i \geq 1$ or $a_i > 0$ but wlog we assume $a_i = 1$ is used.) The cost of changing $a_i = 1$ is 1. Clearly, we have that $\Phi'(I \cup I')$ contains the fraudulent transaction and since H hits every set $s \in S$, for every $s \in S$, there must be at least an attribute a_i in the corresponding tuple of s whose value 0 when the condition $a_i = 1$ according to the new rule Φ'. Hence, it follows that $\Phi'(I \cup I')$ does not contain any tuples in $I \cup I'$ other than the fraudulent tuple.

We show next that the expression $\text{cost}(M) - (\alpha * \Delta F + \beta * \Delta L + \gamma * \Delta R)$ is the minimum possible, assuming that $\alpha = \beta = \gamma > 1$ (β is actually irrelevant here) and the cost of each modification is 1. Suppose there is another set of modifications whose cost is lower than the above. It must be that none of the unlabeled tuples are selected by the modified rules since one can always avoid capturing an unlabeled tuple and lower the cost even further by adding an appropriate condition $a_i = 1$. Furthermore, by the same reason, the fraudulent tuple must be selected by the modified rules. Thus, the expression $\text{cost}(M) - (\alpha * \Delta F + \beta * \Delta L + \gamma * \Delta R)$ is the minimum possible when the number of $a_i = 1$ conditions correspond to the size of a the minimum hitting set. Any smaller value will mean we have a smaller cardinality hitting set which is a contradiction to our assumption that H is a minimum hitting set.

For the converse, let M be the set of modifications made to Φ such that $\text{cost}(M) - (\alpha * \Delta F + \beta * \Delta L + \gamma * \Delta R)$ is minimum. Again, wlog, we may assume that the modifications must be of the form $a_i = 1$.

Let $H = \{x_i \mid a_i = 1 \text{ in the modified rule}\}$. We show next that H is a minimum hitting set. First, we show that H is a hitting set. Suppose not, then there is a set $s \in S$ such that $H \cap s = \emptyset$. Let t be the tuple that corresponds to s in the transaction table. This means that $\Phi'(I \cup I')$ contains t, since t contains the value 1 in every attribute a_i where $a_i = 1$ in the modified rule. Pick an element, say $x_j \in s$ such that $x_j \notin H$. Now if we add the modification $a_j = 1$, the change in cost is +1-γ. Since $\gamma > 1$, we have that the new total cost is lower than the original cost which contradicts the assumption that M is a set of modifications that would give the minimum total cost.

Next, we show that H is a minimum hitting set. Suppose not, then there is another hitting set H' where $|H'| < |H|$. With H', it is straightforward to construct a set of modifications whose cost is lower than $\text{cost}(M) - (\alpha * \Delta F + \beta * \Delta L + \gamma * \Delta R)$.

In our running example, a modified rule is

$$A_1 \leq \top \wedge A_2 = 1 \wedge A_3 \leq \top \wedge A_4 = 1 \wedge A_5 \leq \top$$

since a minimum hitting set is $\{A_2, A_4\}$. □

The reduction of the above proof shows that the NP-hardness result may arise because we allow the size of the schema of the transaction relation to vary. We show next that, even if we fix the size of the schema, the NP-hardness result continues to hold.

THEOREM 4.3. *The* RULE GENERALIZATION PROBLEM *is NP-hard even if Φ is perfect for I and the size of the schema of the transaction relation is fixed.*

SKETCH. The proof makes use of a reduction from the set cover problem and in the reduction, a single unary transaction relation is used. We build a taxonomy of the elements of a set cover instance according to which element belongs to which set. The relation is initially empty and assume Φ is initially empty as well. The cost of adding rule with a condition is 1 and we assume that the cost of adding the condition $A \leq \top$ is very high (i.e., it is prohibited). The new transaction relation I' consists of n new fraudulent transactions, one for each element of the universe in the set cover instance. One can then establish that a set of rules of minimum cost can be derived from a minimum set cover and vice versa. Intuitively, each rule has the form $A \leq S_i$ where each S_i is part of the solution to the instance of the minimum set cover problem. □

The Algorithm In view of the hardness result, we develop a heuristic algorithm (Algo. 1) for determining a best set of generalizations to capture a given set of fraudulent transactions.

In the algorithm, we use I to denote both old and new transactions. Observe that one reason for the hardness of the rule generalization problem comes from the desire to identify a set of modifications that captures all fraudulent transactions and is *globally optimal*. Instead, our heuristic works in a greedy manner, gradually covering more and more uncaptured transactions. Rather than treating each transaction individually, we split the fraudulent transactions into smaller groups (clusters) of transactions that are similar to each other, based on a distance function, and treat each cluster individually. We denote the set of clusters by C. Each cluster in C is represented by a *representative tuple*. Intuitively, a representative tuple of a cluster is a tuple that "contains" every tuple in that cluster. Hence, if a rule is generalized to capture the representative tuple, then it must also capture every tuple in the associated cluster. The algorithm then identifies for each representative tuple the (top-k) best rule modifications to capture it. The proposed modifications are verified with the domain expert, who may interactively further adapt them or ask for additional suggestions. Note that the modifications made to the rules by the algorithm may result in capturing some legitimate tuples. We will see how this too may be avoided later.

We next describe our algorithm more formally. We first define each of the components, then provide a comprehensive example that illustrates all.

Representative tuple of a cluster The *representative tuple* f of a cluster C is a tuple with the same schema as tuples in C such that for every attribute A, $f.A$ contains $t.A$ for every $t \in C$. If A is an attribute with a total order, then $f.A$ is an interval that contains $t.A$.

Algorithm 1: Generalize rules to capture new fraudulent tuples

Input: A set Φ of rules for a transaction relation I (contains old and new transactions), with $F \subseteq I$ and $F' \subseteq I$.
Output: A new set Φ' of rules that captures $F \cup F'$.

1 Let C denote the result of clustering tuples in $F \cup F'$.
2 **foreach** $C \in C$ **do**
3 Let $f(C)$ be the representative tuple of C.
4 Let Top-$k(f(C))$ denote the top-k rules for $f(C)$ based on Equation 2.
5 **foreach** $C \in C$ **do**
6 **while** *there does not exist a rule* r *such that* $f(C) \in r(I)$ **do**
7 **if** *Top-*$k(f(C))$ *is non-empty* **then**
8 Remove the next top rule r from Top-$k(f(C))$.
9 Construct the smallest generalization of r to r' so that $f(C) \in r'(I)$.
10 Ask whether the rule r' is correct.
11 **if** *the domain expert agrees with the modified* r' **then**
12 Replace r with r' in Φ.
13 **else**
14 Ask the domain expert which modifications in r' are undesired.
15 Revert the modifications to the original conditions of r as indicated by the domain expert.
16 Allow the domain expert to make further generalizations to the proposed rule.
17 **else**
18 Create a rule that will select exactly $f(C)$ and add it to Φ.
19 **return** Φ as Φ'.

If A is a categorical attribute, then $f.A$ is a concept that contains $t.A$ for every $t \in C$. Furthermore, $f.A$ is the smallest interval (resp. concept) that has the above property.[2] In other words, f is the "smallest" such tuple that contains every tuple in C. Intuitively, the clustering step, which generates representative tuples, provides a higher-level semantic abstraction to the fraudulent tuples that are to be captured.

Distance of a rule from a representative tuple The notation $|f - r|$ denotes the *distance* of a rule r from a representative tuple f. Intuitively it reflects how much the conditions in the rule need to be generalized for the rule to capture the representative tuple. It is formally defined as:

$$\Sigma_i^n (f.A_i - r.A_i), \tag{1}$$

where $r.A_i$ denotes the interval or concept associated with the condition of attribute A_i in the rule r, and n is the arity of the transaction relation.

The distance between two attribute intervals is defined as follows. If $f.A$ is the interval $[s_1, e_1]$ and $r.A$ is the interval $[s_2, e_2]$, then $|[s_1, e_1] - [s_2, e_2]|$ is the sum of sizes of the smallest interval(s) needed to extend $[s_2, e_2]$ so that it contains $[s_1, e_1]$. For example, the distance of $|[1, 5] - [5, 100]|$ is 4, while the distance of $|[1, 100] - [1, 5]|$ is 95. The distance of $|[5, 10] - [1, 100]|$ is 0, since [1,100] already covers [5,10]. If an attribute A is categorical, then $|f.A - r.A|$ is the length of the smallest "ontological distance" that need to be added to $r.A$ so that it contains $f.A$. For example, |Offline with PIN - Online with CCV| is 1, and |Offline without PIN - Online with CCV| is 2. By leveraging concepts in the ontology when available, the resulting rules have a more meaningful interpretation.

The overall cost function The overall cost of modifying a rule r to capture a representative tuple f then reflects the amount of modifications that need to be carried (Equation 1) minus the benefit derived from those modifications:

$$\Sigma_i^n (f.A_i - r.A_i) - (\alpha * \Delta F + \beta * \Delta L + \gamma * \Delta R) \tag{2}$$

Putting things together We now have all the ingredients for describing our algorithms. The algorithm proceeds by clustering the transactions into groups and then computes a representative tuple for each cluster. [3] For every cluster C, we compute its representative tuple $f(C)$ as well the cost (according to Equation 2) of modifying each of the rules to capture it, and select the k rules with minimal cost. We refer to them as the top-k rules (see Line 4 of Algo. 1). In Line 8, we pick the top rule in top-$k(f(C))$. If the rule $r(I)$ does not already contain $f(C)$, we will attempt to make the smallest generalization on r to r' so that $r'(I)$ contains $f(C)$. Whenever the interval or concept of an attribute $r.A$ does not contain $f.A$, we will attempt to modify $r.A$ by computing the smallest extension needed on $r.A$ based on its distance from $f.A$. We perform this extension on every attribute A of r where $r.A$ does not contain $f.A$. This is what we mean by "generalize r minimally to r' so that $f(C) \in r'(I)$" in line 9.

Next, we proceed to verify the new rule r' with our potential modifications with the domain expert. If the domain expert agrees with the proposed modifications (lines 11,12), we will replace r with the new rule r' in Φ. Otherwise, we will refine our question to probe the domain expert further on whether or not there are parts of the modifications that are desired even though the entire rule r' is not what the domain expert desires (lines 14,15). We then modify only the desired modifications, if any. The next step allows the domain expert to make further generalizations to the rules. After this, the algorithm proceeds to pick another closest rule to $f(C)$ to attempt to capture $f(C)$. Line 18 captures the case when we ran out of rules to modify. If this happens, we will construct a new rule to cover f by stating the exact conditions that are required.

We conclude with a remark regarding the computational complexity of the algorithm. All components of the algorithm (i.e., clustering, computation of representative tuples for each cluster and the top-k rules) execute in PTIME in the size of its input. Hence each iteration executes in PTIME in the size of the input. The number of iterations per cluster is dependent on the amount of refinements that the expert makes to the suggested rule modifications (shown in our experiments to be fairly small).

Example 4.4. The relation below depicts the representative tuples of the clusters formed from the six fraudulent transactions from Figure 2. The first tuple is the representative tuple of the cluster that consists of the first two tuples in Figure 2. The second (resp. third) tuple below is the representative of the cluster that consists of only the 4th tuple in Figure 2 (resp., 6th, 7th, and 8th tuples).

[2] If there are multiple such concepts, e.g. in non tree-shaped concept hierarchies, we pick one.

[3] In our implementation, we use the clustering algorithms of [12], but other clustering algorithms can be similarly be used.

Representatives of fraudulent transactions in Figure 2:

Time	Amount	Transaction Type	Location
[18:02,18:03]	*[106,107]*	*Online, no CCV*	*Online Store*
[19:08,19:08]	*[114,114]*	*Online, no CCV*	*Online Store*
[20:53,20:58]	*[44,48]*	*Offline, without PIN*	*GAS Station B*
:	*:*	*:*	*:*

Consider a domain expert, Elena, that is working with the system. The first rule in Figure 1 is the closest to the first representative tuple above. This is because Equation 2 evaluates to (0+4+0+0)-(2+0+0)=2 for the first rule and the first representative tuple, whereas the second and third rule of Figure 1 and the first representative tuple have scores (53+4+0+0)-(2+0-1)=56 and (178+0+0+1)-(6+0-3)=168, respectively, and are thus ranked lower than the first rule. The number '1' in the last calculation denotes the ontological distance between "Gas Station A" and "Gas Station B". Since they are both contained under "Gas Station", the distance is 1.

Algo. 1 will thus propose to modify the condition of the first rule from "Amt ≥ 110" to "Amt ≥ 106" to capture the representative tuple. It then proceeds to verify the modification with Elena. Suppose Elena accepts the proposed modification but further generalizes the condition rounding it down to "Amt ≥ 100" instead. So the new rule 1 is

1) *Time ∈ [18:00,18:05] ∧ Amt ≥ 100.*

Besides the fact that rounded values may be preferred by domain experts over more specific values, such rounding may embody domain-specific knowledge that may possibly lead to the discovery of more fraudulent transactions, particularly from transactions that are unlabeled, or the discovery of legitimate transactions that should not have been labeled as legitimate.

For the second and third cluster, similar interactions occur between RUDOLF and Elena. The new rules that result are:

2) *Time ∈ [18:55,19:15] ∧ Amt ≥ 110.*

3) *Time ∈ [20:45, 21:15] ∧ Amt ≥ 40 ∧ Location≤Gas station'.*

To conclude, observe that Algo. 1 allows Elena to make further generalizations to the rules. Elena rounded the value down from 106 to 100 because her experience tells her that if frauds occur with amount greater than $106, then it is likely to occur a few dollars below $106 as well. Hence, she generalized (i.e., rounded down) the value to $100. In making such generalizations, the fraudulent transactions will continue to be captured. However, the modified rules may now capture (more) non-fraudulent transactions. Nonetheless, we still allow such generalizations since these are deliberate changes made by Elena, the domain expert. More typically, however, such "rounding generalizations" tend to be meaningful generalizations that may lead to the discovery of more fraudulent transactions (i.e., unlabeled transactions that should be classified as fraudulent or legitimate transactions that are mistakenly labeled as legitimate)[4]. As we shall show in Example 4.7, Elena can also leverage her experience or domain knowledge to pinpoint the right conditions for avoiding legitimate transactions.

We describe next how over-generalization may be treated.

4.2 Rule Specialization Algorithm

In the previous subsection, we have seen how one generalizes rules to capture fraudulent transactions. We now discuss the opposite case, where we wish to specialize rules instead, in order to exclude legitimate transactions when there no new fraudulent transactions or unlabeled transactions but there are new legitimate transactions.

[4]This is from our conversations with domain experts on credit card fraud detection.

We call this special case the RULE SPECIALIZATION PROBLEM. Here again we can show hardness results analogous to Theorem 4.1 and 4.3.

THEOREM 4.5. *The* RULE SPECIALIZATION PROBLEM *is NP-hard even if Φ is perfect for I. The problem is NP-hard even when I contains only unlabeled transactions and I' consists of only one legitimate transaction.*

PROOF. Given an instance of the hitting set problem, we construct an instance of the RULE SPECIALIZATION PROBLEM as follows.

The transaction relation has $|U|$ columns, one for each element in U. For every set s in S, we construct a characteristic tuple of s by placing a 0 in position i if $x_i \in s$ and 1 otherwise. Hence there are $|S|$ tuples in the transaction relation I so far and the fraudulent transactions $F = I$. The set Φ consists of a single rule

$$A_1 \leq \top \wedge \ldots \wedge A_{|U|} \leq \top,$$

where $\top$ denotes the top element, and hence, $\Phi(I)$ currently captures all fraudulent transactions F. The new transaction relation I' consists of a single tuple (1,1,1,...,1). This set L' of legitimate transactions is a singleton set consisting of only (1,1,1,...,1). That is, $L' = I'$. This is the legitimate transaction that we wish to exclude.

With $F' = L = \emptyset$, our goal is to specialize the rule in Φ to capture exactly the fraudulent tuples F only. Like in the proof of Theorem 4.1, we assume that each modification is associated with a unit cost and $\alpha = \beta = \gamma > 1$.

As an example, consider the same hitting set as in the proof of Theorem 4.1, where $U = \{A_1, A_2, A_3, A_4, A_5\}$ and $S = \{s_1, s_2, s_3\}$, where $s_1 = \{A_1, A_2, A_3\}$, $s_2 = \{A_2, A_3, A_4, A_5\}$, and $s_3 = \{A_4, A_5\}$. The transaction relation $I \cup I'$ is shown below (where I' is shown in gray). The last column annotates the type of tuple (i.e., F for tuples in F and L' for tuples in L').

A_1	A_2	A_3	A_4	A_5	
0	0	0	1	1	F
1	0	0	0	0	F
1	1	1	0	0	F
1	1	1	1	1	L'

It is straightforward to verify that the reduction to an instance of the RULE SPECIALIZATION PROBLEM can be achieved in polynomial time in the size of the input of the Hitting Set Problem. We now show that a solution for the Minimum Hitting Set Problem implies a solution for the RULE SPECIALIZATION PROBLEM and vice versa.

Let H be a minimum hitting set. For every element x_i in H, we duplicate the original rule (except if this is the last element in H) and modify the corresponding condition to $a_i = 0$ in the copy of the rule. (The condition can also be $a_i \leq 0$ or $a_i < 1$ but wlog we assume $a_i = 0$ is used.) Recall that the cost of changing $a_i = 0$ is 1, and the cost of duplicating a rule is 1. Clearly, we have that $\Phi'(I \cup I')$ contains F. Indeed, since H hits every set $s \in S$, there must be an element in s whose corresponding value under an attribute a_i is 0 when the condition $a_i = 0$ according to a new rule in Φ'. Hence, it follows that $\Phi'(I \cup I')$ contains F and since each rule in Φ' contains a condition of the form $a_i = 0$, the legitimate transaction (1,1,1,...1) will not be among $\Phi'(I \cup I')$.

We show next that the expression $\text{cost}(M) - (\alpha * \Delta F + \beta * \Delta L + \gamma * \Delta R)$ is the minimum possible. Suppose there is another set M' of modifications to Φ'' such that $\text{cost}(M') - (\alpha * \Delta F + \beta * \Delta L + \gamma * \Delta R)$ is less than the previous expression. Observe that every rule in Φ'' must contain at least a condition that is specific to selecting

a fraudulent transaction. That is, for every rule, $a_i = 0$ for some i since otherwise, the rule is either redundant or the legitimate transaction will be selected. Also, we can assume that every other condition in the rule in Φ'' cannot contain a condition that selects 1s (e.g., of the form $a_i = 1$). If a rule r contains a condition $a_i = 1$, then we can omit this condition and assume it is $a_i \leq \top$ instead. The rule with $a_i \leq \top$ captures all tuples that are captured by r (and possibly more) and hence, we will continue to capture all fraudulent tuples and continue to exclude the legitimate tuple under this assumption. Similarly, if a rule r contains multiple conditions $a_i = 0$s, then we can omit all but one of the $a_i = 0$s and assume the rest are $a_i \leq \top$. We can now construct a hitting set from M' that is smaller than H, which is a contradiction.

For the converse, let M be the set of modifications made to Φ such that $F \subseteq \Phi'(I \cup I')$, the legitimate transaction l is such that $l \notin \Phi'(I \cup I')$, and $\text{cost}(M) - (\alpha * \Delta F + \beta * \Delta L + \gamma * \Delta R)$ is minimum. As before, observe that each rule must contain at least one modification of the form $a_i = 0$ for some i so that l is not selected. Furthermore, it is easy to see that each rule must contain exactly one such condition $a_i = 0$ only as additional conditions such as $a_j = 1$ or $a_j = 0$ are redundant and can only increase the cost.

Let $H = \{x_i \mid a_i = 0 \text{ in any of the modified rules}\}$. We show next that H is a minimum hitting set. First, we show that H is a hitting set. Suppose not, then there is a set $s \in S$ such that $H \cap s = \emptyset$. In other words, for every element $x_i \in s$, there does not exist a rule in Φ' where $a_i = 0$. Let f be the tuple that corresponds to s in the transaction table. This means that $f \notin \Phi'(I \cup I')$, which contradicts our assumption that $F \subseteq \Phi'(I \cup I')$.

Next, we show that H is a minimum hitting set. Suppose not, then there is another hitting set H' where $|H'| < |H|$. With H', it is straightforward to construct a set of modifications whose cost is lower than M's cost.

In our running example, Φ' contains two rules:

$$A_1 \leq \top \wedge A_2 = 0 \wedge A_3 \leq \top \wedge A_4 \leq \top \wedge A_5 \leq \top$$
$$A_1 \leq \top \wedge A_2 \leq \top \wedge A_3 \leq \top \wedge A_4 = 0 \wedge A_5 \leq \top$$

since a minimum hitting set is $\{A_2, A_4\}$. □

Similarly, we can show that the NP-hardness continues to hold even if we fix the size of the schema.

THEOREM 4.6. *The* RULE SPECIALIZATION PROBLEM *is also NP-hard even if Φ is perfect for I and the size of the schema of the transaction relation is fixed.*

SKETCH. The proof of the above result is similar to that of Theorem 4.3. It makes use of a reduction from the set cover problem and in the reduction, a single unary transaction relation is used. We build a taxonomy of the elements of a set cover instance according to which element belongs to which set. The relation initially contains all elements of the universe of the set cover instance and these transactions are all fraudulent. The set Φ consists of a single rule $A \leq \top$ which captures all fraudulent transactions. The cost of adding a rule and modifying a condition cost 1 each. The new transaction relation I' consists of a single legitimate tuple whose value does not occur among the existing values. One can then establish that a set of rules of minimum cost can be derived from a minimum set cover and vice versa. Intuitively, each rule has the form $A \leq S_i$ where each S_i is part of the solution to the instance of the minimum set cover problem. □

Algorithm 2: Adapt rules to exclude legitimate tuples

Input: A set Φ of rules for a transaction relation I (contains old and new transactions) with $L \subseteq I$ and $L' \subseteq I$.
Output: A new set Φ' of rules that excludes $L \cup L'$.

```
foreach l ∈ (L ∪ L') do
    Let Ω_l = {r ∈ Φ | l ∈ r(I)}.
    foreach r ∈ Ω_l do
        repeat
            Let A be an attribute that has not been considered
              before and where splitting on A to exclude l.A will
              minimize the cost associated with splitting on A.
            Suppose the existing condition on A is A ∈ [b, e].
            Split r into r_1 and r_2 on A as follows:
            Let r_1 be a copy of r except that the condition on A
              is A ∈ [b, prev(r.A)].
            Let r_2 be a copy of r except that the condition on A
              is A ∈ [succ(r.A), e].
            Ask the domain expert whether the split into r_1 and
              r_2 is correct.
            if the domain expert agrees with the modification
              then
                Add r_1 and r_2 to Φ.
                Allow the domain expert to make further
                  modifications to the proposed rules.
                Break out of repeat loop.
        until all attributes have been considered;
        Remove r from Φ.
return Φ as Φ'.
```

The Algorithm In view of the hardness results, we develop a heuristic algorithm (Algo. 2) that greedily determines the best attribute to "split" to avoid capturing each legitimate tuple.

In Algo. 2, we use I to denote both old and new transactions. For each legitimate transaction l in $(L \cup L')$, we determine the set Ω_l of rules that will capture l and modify every rule in Ω_l to ensure that the modified rules will no longer capture l. For a rule r in Ω_l, the algorithm proceeds to pick an attribute A where we can split the condition of the attribute to exclude the value $l.A$. The attribute A that we pick is the one that maximizes the benefit according to the benefit $\alpha * \Delta F + \beta * \Delta L + \gamma * \Delta R$, assuming a fixed cost of modification where we copy the rule and split on the attribute. If there are multiple attributes with the same maximum benefit, we randomly pick one of them. Observe that the heuristic of greedily selecting an attribute that will maximize benefit may not be globally optimal in the end. In the proof of Theorem 4.5, this greedy heuristic is analogous to the strategy of repeatedly picking the attribute (and splitting on the attribute) that will "hit" the most sets until all sets are hit.

Splitting on attributes Once an attribute is selected, the rule is duplicated into r_1 and r_2 and the condition on A in both rules is modified to exclude $l.A$. Observe that since $r(I)$ captures l, we must have that $l.A$ satisfies the rule r's condition on A. In the split, r_1's condition on A accept values from b to the element that is the predecessor of $l.A$, where b denotes the smallest value accepted by the existing condition of r on A. The rule r_2 selects only elements from the successor element of $l.A$ to the largest value (i.e., e) accepted by the existing condition of r on A.

For domains that are discrete and has a total order, the above procedure, $\text{prev}(r.A)$ and $\text{succ}(r.A)$ are well-defined. However, when domains are categorical and has only a partial order, the rules will be split according to the partial order. Let O denote the

set of all concepts (excluding $l.A$) that are parents of $\perp$ in the partial order (i.e., the leaf nodes of the partial order excluding $\perp$). To exclude $l.A$, the algorithm considers how to select a minimum set of concepts to "cover" all concepts in O that excludes $l.A$ at the same time. It can be shown that the problem of computing such a minimum set is analogous to computing a minimum set cover for O. Our procedure adopts the greedy heuristic where we greedily pick a concept in the partial hierarchy that covers the most number of uncovered concepts in O until all nodes in O are covered. For categorical attributes, it may be necessary to duplicate r more than twice, where there is a rule to select each concept in the cover. For example, referring to Figure 1, to exclude "Online, with CCV", we may pick "Offline" and "Online, without CCV" to cover the remaining concepts that are parents of "None". Observe that similar to our algorithm on rule generalization, our rule specialization algorithm also makes use of the ontology whenever available to split the attributes meaningfully. In this case, attributes are split into meaningful concepts in the "lower-level" according to the ontology.

After this, we ask whether the domain expert agrees with the split. If the domain expert agrees, we add both rules r_1 and r_2 to Φ. The domain expert can also add further modifications to the rules (line 13), such as excluding more values than what is suggested by the algorithm, and we break out of the repeat loop. Otherwise, we repeat the loop to attempt to split r on another attribute to avoid capturing l. Note that since l has to be excluded, one of the splits must be deemed correct by the domain expert. After this, we remove r from Φ and repeat the same procedure to modify Φ to exclude the selection of another legitimate transactions.

Example 4.7. We will now illustrate Algo. 2. The legitimate transactions from Figure 2 that are captured by the modified rules of Example 4.4 are shown below for convenience.

	Time	Amount	Transaction Type	Location
l_1	*18:04*	*112*	*Online, with CCV*	*Online Store*
l_2	*19:10*	*117*	*Online, with CCV*	*Online Store*
l_3	*21:01*	*49*	*Offline, with PIN*	*GAS Station A*
	:	*:*	*:*	*:*

Modified rules Φ from Example 4.4:

1) *Time* $\in$ *[18:00,18:05]* $\wedge$ *Amt* $\geq$ *100.*

2) *Time* $\in$ *[18:55,19:15]* $\wedge$ *Amt* $\geq$ *110.*

3) *Time* $\in$ *[20:45, 21:30]* $\wedge$ *Amt* $\geq$ *40* $\wedge$ *Location*$\leq$*Gas station'.*

We would like to adapt the rules to exclude these legitimate transactions (and still continue capture fraudulent transactions). For this example, assume that $\alpha = \beta = \gamma = 1$.

The algorithm considers every legitimate transaction. Since l_1 is caught by rule (1) above, Algo. 2 proceeds to determine which attribute of the rule to split in order to exclude l_1. Splitting on `time` or `amount` or `type` will result in the same maximum benefit: (1*0 (zero unlabeled transactions on either day) + 1*0 (the number of fraudulent transactions that are caught remains unchanged) + 1*1 (one less legitimate transaction that is caught)). Splitting on the attribute `location`, however, will cause additional fraudulent transactions (i.e., the first two transactions in Figure 2) to be missed and hence has a lower benefit than the rest of the attributes.

Suppose the algorithm proposes to split on `time` (an arbitrary choice among `time`, `amount` or `type`). This will result in two rules that will capture all fraudulent transactions that were previously caught by the rule and exclude l_1 at the same time.

r_{11}*: Time* $\in$ *[18:00,18:03]* $\wedge$ *Amt* $\geq$ *100.*

r_{12}*: Time* $\in$ *[18:05,18:05]* $\wedge$ *Amt* $\geq$ *100.*

At this point, Elena can accept this proposal or ask for alternatives. For the purpose of illustrating our algorithm, suppose Elena asked for an alternative proposed modification. Our algorithm may now propose to split on `type` instead. Since there is currently no condition on `type` in the first rule, the condition is implicitly "type $\leq \top$". And because the concepts "Offline" and "Online, without CCV" cover all possible type values (i.e., values immediately above the "None" node in Figure 1) except "Online,with CCV" which we wish to exclude, we have the following two rules:

r_{11}*: Time* $\in$ *[18:00,18:05]* $\wedge$ *Amt* $\geq$ *100* $\wedge$ *Type* $\leq$ *Offline.*

r_{12}*: Time* $\in$ *[18:00,18:05]* $\wedge$ *Amt* $\geq$ *100* $\wedge$ *Type* $\leq$ *Onl.,no CCV.*

Using domain knowledge that only online purchases, especially those without CCVs are of concern, Elena eliminates the rule r_{11}.

After this, our algorithm proceeds in a manner that is similar to what was described before to split the second rule of Φ to omit l_2 (and similarly, the third rule of Φ for l_3). We omit the details here but show the final rules that are obtained.

r_{22}*: Time* $\in$ *[18:55,19:15]* $\wedge$ *Amt* $\geq$ *100* $\wedge$ *Type* $\leq$ *Onl., no CCV.*

r_{31}*: Time* $\in$ *[20:45,21:15]* $\wedge$ *Amt* $\geq$ *40* $\wedge$ *Location* $\leq$ *Gas Station* $\wedge$ *Type* $\leq$ *Online.*

r_{32}*: Time* $\in$ *[20:45,21:15]* $\wedge$ *Amt* $\geq$ *40* $\wedge$ *Location* $\leq$ *Gas Station* $\wedge$ *Type* $\leq$ *No code.*

Observe that whenever a condition is generalized (in Algo. 1), more legitimate or unlabeled tuples may be inadvertently captured by the rule. Hence, further refinements may be needed to tune the rules to a desired state. Conversely, if a condition is specialized, some fraudulent tuples may be inadvertently omitted. Hence, further refinements may be needed to tune the rules to a desired state. As we shall describe next, the rules are interactively refined based on the input of a domain expert such as Elena. In particular, there may be several rounds of refinements through generalizations and specializations before a desired set of rules is obtained.

5 IMPLEMENTATION AND EXPERIMENTS

RUDOLF is implemented in PHP/JavaScript and uses MySQL as the DB engine. Detailed system architecture described in [11].

Datasets We have access to a real-world datasets of credit card transactions by a financial company XYZ[5]. Due to the sensitivity of credit card-related information, we used anonymized version of the dataset. The dataset consists of transaction sets of various sizes from 15 financial institutes (FIs) for the first quarter of 2016. Each transaction set varies from 100K to 10M transactions and most of them consists of about 500K transactions. The percentage of fraudulent transactions varies between 0.5% to 2.5% between different FIs. The number of missclassified transactions (i.e., fraudulent transactions that are marked as legitimate and vice-versa) varies between 35% and 50%. The transactions contain both numerical (time, amount, number of previous actions, etc.) and also categorical (location, client type, etc) data. Along with the transactions, we obtained 15 rules-sets, one for each of the 15 FIs for the same time period from company XYZ. We also obtained the change history and versions of those rules. A small FI typically has about 10 rules while a big FI typically has about 130 rules. Most FIs have about 55 rules on average. Each time the rules are modified, the rules undergo about 10 rounds of modifications on average. The transactions in the data sets are annotated as fraudulent/legitimate, and we take these annotations as the ground truth. Each transaction also has a risk score, which is a value between 0 and 1000, that is generated by the company's machine learning algorithm to determine the chance that the transaction is fraudulent. The

[5]Name omitted per company request

fraudulent transactions can be captured by the set of rules given by the company and allowing the users to refine the rules over time. Another option is to apply a rule that classify all transactions with risk score above a certain threshold as fradulent.

Ontology In the experiments for the location attributes we used a geographical ontology (containing different relations, e.g., capital city, located in, region, continent, etc) that was built semi-automatically (using DBPedia [13]) and manually verified by the domain experts.

Experiment scenarios The different sizes of transaction sets allowed us to vary our experiments with different dataset sizes (from 100K to 10M, with the average value being 500K).

We run each experiment with 8 users (fraud detection experts from company XYZ) and as the variance was less that 2% we present here the average. We also ran our experiments with 10 student volunteers to determine whether the level of expertise affects the results. To simulate the work of a domain expert, we we spilt each dataset into two parts of approximately the same size, before and after a certain point in time. We advanced in time from this point and examined, at different points in time, how the expert adapts the rules in response to transactions arriving up to that point.

We compared the performance of RUDOLF to three alternative solutions, to be detailed below. For each of the algorithms and each of the datasets, we varied the number of new transactions arriving between consecutive rounds of rule refinement. The number of new transactions varies from 10% to 20% of the dataset, with the default being 10%, and this corresponds closely to what happens in real-life between rounds of rule refinement.

Baseline algorithms We consider the two extreme baselines: A *fully-manual* setting, where rules are manually refined by experts without the help of the system (the current setting that is used by company XYZ experts in their daily work), and a *fully-automatic* setting that uses the risk score produced by the ML algorithm and a single rule that selects fraudulent transactions based on their risk scores. Observe that this algorithm essentially generates a single new rule of the form *score greater than threshold* (rather than refining an existing set of rules). We also compared to the baseline algorithm *No Change*, which denotes the given rules without any changes.

Observe that the fully manual setting is arguably our "toughest" competitor since the rules are modified by experts and the experts are not limited by any time constraint to refine the rules.

In addition to the above, we also consider a variant of RUDOLF, denoted RUDOLF$^-$, that automatically refines the existing set of rules by accepting the modifications proposed by the system without consulting an expert. We also considered RUDOLF -s, which is the system RUDOLF that does not refine categorical attributes (and hence does not use ontologies) of rules. To the best of our knowledge, all existing systems refine only numerical attributes of rules. Hence the performance of RUDOLF -s will allow us to understand how RUDOLF compares with systems that are only restricted to refine numerical attributes. In fact, we discovered that RUDOLF -s gives almost same results as the fully-manual system and also RUDOLF$^-$. Hence, we omit the results of RUDOLF -s completely.

Measurements In our experiments, we measured the efficiency of the algorithms in terms of the effectiveness of the derived rules and the amount of time that the domain experts saved as a consequence of using our system. We also measured the running time required by RUDOLF to select the proposed modifications. For our datasets this was always at most one second, and we thus omit the exact measures.

To measure the effectiveness of a set of rules derived by any of the methods, we consider its *prediction quality*, namely how correctly it identifies future frauds. For that, we examine the set of transactions from the given point in time where the rules were derived and until the end of the dataset. For these future transactions we count, for each set of rules, the percentage out of all fraudulent (resp. legitimate) transactions that it identifies (resp. wrongly classifies as fraudulent).

To understand of how many modifications each method entailed, we also computed the cumulative number of rule updates that each method required. We measure this only for RUDOLF, RUDOLF$^-$, and the fully-manual methods, which directly change existing rules.

Finally, we measured the time the experts took to refine the rules.

Results We first report on our experiments with the domain experts. Our first experiment examines the performance of the algorithms as time advances, with all parameters set to their default value. As explained above, at each point in time, (i.e. after a certain percentage of the transactions has been observed), the algorithms are invoked to derive a corresponding updated set of rules. Figure 3(a) shows the (cumulative) number of modifications that RUDOLF, fully-manual and RUDOLF$^-$ method performed to the rules. We see that RUDOLF performs less modifications than its competitors.

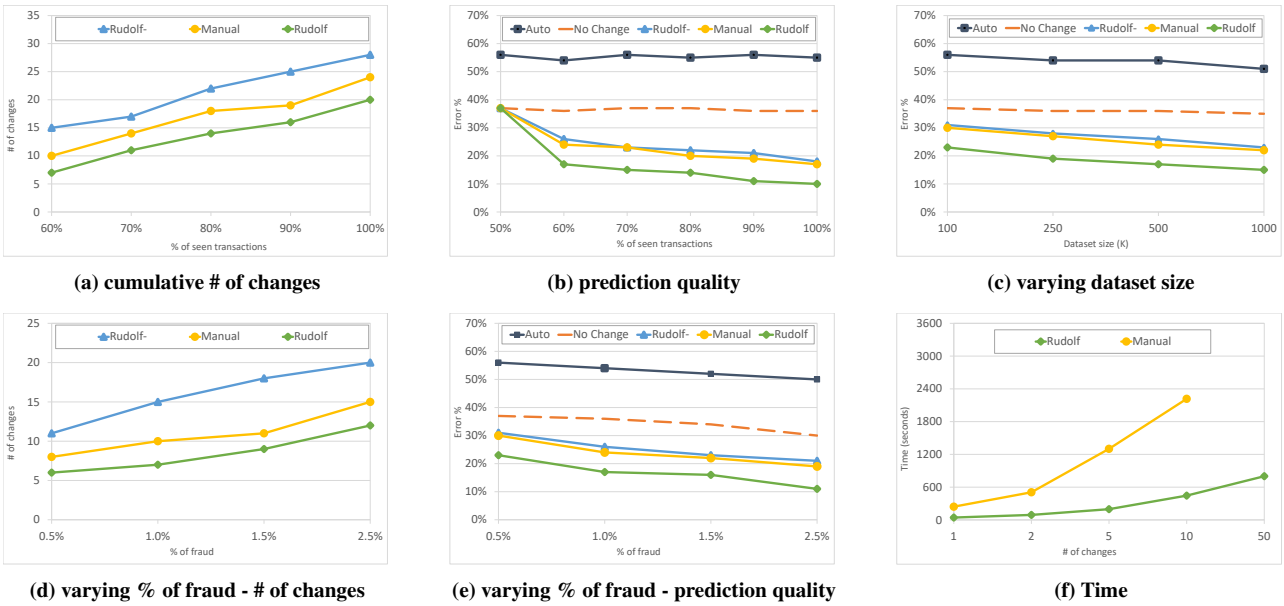
We can see this more clearly in Figure 3(b), which illustrates the prediction quality of the derived sets of rules, in terms of the percentage of misclassified future transactions (lower percentage of error implies better prediction quality). RUDOLF performs the best, providing the best prediction. The fully manual rule derivation provides less accurate predictions, though still better than the two automatic competitors. Among the two, RUDOLF$^-$, that incrementally refines the rules, still performs better than the threshold-based ML approach.

We note that the difference in performance between RUDOLF$^-$ and RUDOLF demonstrates the importance of incorporating experts and their domain knowledge in the loop.

For the experiments above, the rules were periodically refined in hops of 10% of the transactions. For different hops sizes, the results are also similar, except that convergence naturally arrives after fewer (proportionally) iterations for larger hops.

Our next experiment examines the performance of the algorithms for varying dataset, with almost the same percentage of fraud, but different sizes. The size had no significant effect on the number of rule modifications performed by the algorithms, but the prediction quality slightly improved as more data was available. Figure 3(c) illustrate, for varying dataset sizes, the prediction quality of the rules after the first refinement round, in terms of the percentage of misclassified transaction. Here again, lower percentage means better quality. As before, RUDOLF yields best results. We can see that the error of all algorithms slightly decreases as the size of the data set grows. The improvement is only small as fraudulent transactions of the existing fraud patterns are distributed throughout the datasets, so the additional data reveals some, but not huge, amount of new information. Similar results were obtained for the following refinements rounds and we thus omit the graphs.

Next, we examine the performance of the algorithms for varying percentages of fraudulent transactions. We took 4 different



(a) cumulative # of changes **(b) prediction quality** **(c) varying dataset size**

(d) varying % of fraud - # of changes **(e) varying % of fraud - prediction quality** **(f) Time**

Figure 3: Experimental results

customers databases of roughly the same size, but different fraud percentages (0.5% to 2.5%). All other parameters are set to their default values. Figures 3(d) 3(e) show, respectively, the number of rule updates and percentage of error after the first refinement round. We can see that an increased number of fraudulent transaction entails more rule modifications to capture them. The classification error slightly increases with more fraudulent transactions, but here again RUDOLF achieves the lowest error.

Finally we note that rule refinement with RUDOLF not only consistently yielded superior fraud prediction, but also reduced the time required from the experts by a factor between 4 to 5. (Around 50 seconds per round for RUDOLF compared to 4-5 minutes without). We measured time of our experts performance, depicted in figure 3(f). We asked them to fix up to 50 problematic transactions in both manual and automatic way. Interestingly, no expert finished all 50 fixes in the manual mode (a well-trained expert from company XYZ usually can fix 30-40 transactions per work-day). The fact that RUDOLF leads in performance across all parameters is interesting as the rules derived manually by experts are typically considered as ground truth and yet, RUDOLF is able to do better (i.e., with less changes, and with lower percentage of error) with less data. This was consistent in all the experiments. Furthermore, all users reported that working with RUDOLF was convenient and effective in the sense that the rules/modifications proposed by the system helped them identify and focus on the problematic rules and the needed treatment. To conclude we observed that around 75% of the modifications were condition refinements, 20% rule splits, and 5% rule addition.

Interestingly, our experiments with novice users (student volunteers) show similar trends. In particular, also for novice users, the rules generated with the assistance of RUDOLF were of best quality and produced much faster than in all alternatives. We omit the graphs for space constraints and only note that as expected, compared to the domain experts, the overall perdition quality, even with RUDOLF , was lower (by about 5%) than for the experts, but still significantly better (by 25%) than what the novice users would have achieved alone.

6 RELATED WORK

The identification of fraudulent transactions is essentially a classification problem. Classification has been a fundamental problem in machine learning and data management [14, 15], and crowdsourcing has recently emerged as a major problem solving paradigm [16]. Many classification works have used crowdsourcing to obtain training data for learning [17–20]. This is complimentary to our work: In RUDOLF the crowd (of experts) is employed to maintain classifications rules, which might have been initially learnt through such training. Besides learning-based models, rules are also used for classification. Most of the previous research in rule-based classifiers focus on how to learn rules from the training data. In contrast, [21] employs both learning and analyst experts that manually create classification rules using regular expressions. In [22] that describes LinkedIn's job title classification system, experts and crowdsourcing are also heavily used. In both cases however the ongoing refinement of rules in a changing environment, which is the focus of RUDOLF, is not considered.

In addition to machine learning-based methods, there are multiple fraud-detection techniques that have been considered. For example [4] uses a decision tree, defined recursively for nodes and edges of the tree and using the ratio between number of transactions that satisfy some condition to label them accordingly. Other methods, e.g. [5], are based on genetic programming, used to classify transactions into suspicious and non-suspicious ones. Another class of the algorithms for fraud detection is based on clustering techniques. An example is [6] that clusters users based on common behavior and then considers as suspicious the transactions that take the user outside its cluster. Bayesian networks are used both to detect fraud in telecommunications (e.g. [23]) and in the credit card industry (e.g. [24]). Neural networks are also used for fraud detection. For instance [25] presents an online fraud detection system, based on a neural classifier. All these techniques are complimentary to ours and can be used to deriving the initial base-set of rules.

Another class of work similar RUDOLF is that of Concept Drifts, which are changes that occur on the distribution of the input that affects the learning system and thus the output. [8] deals with concept drifts by using sliding window that adaptively remembers more or less items from the training set (the closest

past) according to whether it recognized a concept drift or not. Other system ([9]) is classification system based on decision rules. Even though these systems can compute the nearest neighbors for the closest rules and generalization for numerical values, they do not support generalization and specification on categorical attributes, do not involve a human expert in the loop, and do not allow configuration of weight for different kind of errors (false positives and false negatives).

Finally, if we view our transaction relation as the source database and the set of fraudulent transactions as our target database, then the work on deriving queries or schema mappings based on source-target databases (e.g., [26–28]) is also relevant. Similarly, techniques for rule mining and, in particular, inductive logic programming (e.g., [29]) can also be used for fraud detection, where the fraudulent transactions can be seen as positive examples and the legitimate transactions can be seen as negative examples. However, the language of schema mappings and inductive logic programming are different from our rule language and more importantly, the rules derived cannot be interactively adapted.

7 CONCLUSION AND FUTURE WORK

We present RUDOLF, a novel system that assists domain experts in defining and adapting rules in dynamic environments. We show that the problem of identifying the best candidate adaptation for a core language is NP-hard and present PTIME heuristic algorithms for determining the set of rules to adapt and working interactively with the domain experts until they are satisfied with the resulting rules. Our experiments with real-world data sets demonstrate the promise that RUDOLF is an effective and efficient tool for rule refinement.

One direcvtion for future work is the use of more sophisticated cost model. Instead of associating a cost with every modification made to a condition in the rule, one can varying the cost depending on the attribute or even rule that is modified and these costs/weights can be learned or adjusted based on user feedback, satisfaction of the suggested modification etc. Similarly, the parameters α, β and γ used in our cost formula to weight the importance of misclassifying of fraudulent/legitimate/unlabeled transactions may also be dynamically adapted based on such user feedback.

Acknowledgements This work has been partially funded by the European Research Council under the FP7, ERC grant MoDaS, agreement 291071, and by grants from the Blavatnik Cyber Security center and the Israel Innovation Authority. Work was done while Tan was at UCSC. Tan was partially supported by NSF grant IIS-1524382 at UCSC.

REFERENCES

[1] "The us sees more money lost to credit card fraud than the rest of the world combined," http://read.bi/18Gin67.
[2] "Card fraud worlwide," http://nilsonreport.com/publication_chart_and_graphs_archive.php?year=2015.
[3] "How credit card companies spot fraud before you do," http://money.usnews.com/money/personal-finance/articles/2013/07/10/how-credit-card-companies-spot-fraud-before-you-do.
[4] A. I. Kokkinaki, "On atypical database transactions: Identification of probable frauds using machine learning for user profiling," *KDEX*, 1997.
[5] P. J. Bentley, J. Kim, G.-H. Jung, and J.-U. Choi, "Fuzzy darwinian detection of credit card fraud."
[6] R. J. Bolton and D. J. Hand, "Statistical fraud detection: A review," *Statistical Science*, vol. 2002, pp. 235–255, 2002.
[7] A. Chapman and H. V. Jagadish, "Why not?" in *SIGMOD*, 2009, pp. 523–534.
[8] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Machine Learning*, vol. 23, no. 1, pp. 69–101, 1996.
[9] F. J. Ferrer-Troyano, J. S. Aguilar-Ruiz, and J. C. R. Santos, "Data streams classification by incremental rule learning with parameterized generalization."
[10] C. Phua, V. C. S. Lee, K. Smith-Miles, and R. W. Gayler, "A comprehensive survey of data mining-based fraud detection research," 2010.
[11] T. Milo, S. Novgorodov, and W. Tan, "Rudolf: Interactive rule refinement system for fraud detection," *PVLDB*, vol. 9, no. 13, pp. 1465–1468, 2016.
[12] M. Shindler, A. Wong, and A. Meyerson, "Fast and accurate k-means for large datasets," in *NIPS*, 2011, pp. 2375–2383.
[13] "DBPedia," http://dbpedia.org.
[14] T. M. Mitchell, *Machine learning*, ser. McGraw Hill series in computer science. McGraw-Hill, 1997.
[15] R. Ramakrishnan and J. Gehrke, *Database management systems (3rd ed.)*. McGraw-Hill, 2003.
[16] A. Doan, R. Ramakrishnan, and A. Y. Halevy, "Crowdsourcing systems on the world-wide web," *Commun. ACM*, vol. 54, no. 4, pp. 86–96, 2011.
[17] S. Vijayanarasimhan and K. Grauman, "Large-scale live active learning: Training object detectors with crawled data and crowds," in *CVPR*, 2011, pp. 1449–1456.
[18] V. Ambati, S. Vogel, and J. G. Carbonell, "Active learning and crowd-sourcing for machine translation," in *LREC 2010*.
[19] E. Kamar, S. Hacker, and E. Horvitz, "Combining human and machine intelligence in large-scale crowdsourcing," in *AAMAS*, 2012, pp. 467–474.
[20] D. R. Karger, S. Oh, and D. Shah, "Iterative learning for reliable crowdsourcing systems," in *NIPS 2011*.
[21] C. Sun, N. Rampalli, F. Yang, and A. Doan, "Chimera: Large-scale classification using machine learning, rules, and crowdsourcing," *PVLDB*, vol. 7, no. 13.
[22] R. Bekkerman and M. Gavish, "High-precision phrase-based document classification on a modern scale," in *KDD 2011*, 2011, pp. 231–239.
[23] K. J. Ezawa and S. W. Norton, "Constructing bayesian networks to predict uncollectible telecommunications accounts," *Intelligent Systems*, 1996.
[24] S. Maes, K. Tuyls, B. Vanschoenwinkel, and B. Manderick, "Credit card fraud detection using bayesian and neural networks," in *NAISO*, 2002.
[25] D. J.R., G. F., S. C., and C. C.S., "Neural fraud detection in credit card operations," *IEEE Trans. on Neural Networks*, 1997.
[26] A. D. Sarma, A. G. Parameswaran, H. Garcia-Molina, and J. Widom, "Synthesizing view definitions from data," in *ICDT*, 2010.
[27] Q. T. Tran, C. Y. Chan, and S. Parthasarathy, "Query reverse engineering," *VLDB J.*, vol. 23, no. 5, pp. 721–746, 2014.
[28] B. Alexe, B. ten Cate, P. G. Kolaitis, and W. C. Tan, "Designing and refining schema mappings via data examples," in *SIGMOD*, 2011, pp. 133–144.
[29] L. Galárraga, C. Teflioudi, K. Hose, and F. M. Suchanek, "Fast rule mining in ontological knowledge bases with AMIE+," *VLDB J.*, vol. 24, no. 6, 2015.