Rudolf: Interactive Rule Refinement System for Fraud Detection

Tova Milo [1] Slava Novgorodov [1] Wang-Chiew Tan [2]

[1] *Tel-Aviv University* [2] *University of California, Santa Cruz*

[1] {milo, slavanov}@post.tau.ac.il [2] tan@cs.ucsc.edu

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. In addition to machine learning-based techniques, credit card companies often employ domain experts to manually specify rules that exploit domain knowledge for improving the detection process. Over time, however, as new (fraudulent and legitimate) transaction arrive, these rules need to be updated and refined to capture the evolving (fraud and legitimate) activity patterns. The goal of the RUDOLF system that is demonstrated here is to guide and assist domain experts in this challenging task.

RUDOLF automatically determines a best set of candidate adaptations to existing rules to capture all fraudulent transactions and, respectively, omit all legitimate transactions. The proposed modifications can then be further refined by domain experts based on their domain knowledge, and the process can be repeated until the experts are satisfied with the resulting rules. Our experimental results on real-life datasets demonstrate the effectiveness and efficiency of our approach. We showcase RUDOLF with two demonstration scenarios: detecting credit card frauds and network attacks. Our demonstration will engage the VLDB audience by allowing them to play the role of a security expert, a credit card fraudster, or a network attacker.

1. INTRODUCTION

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.

In this demo, we present RUDOLF, a system that is capable of assisting domain experts to define and refine rules for fraud detection. The system addresses a real problem that is faced by a credit card company whose name we cannot disclose. Everyday, the company receives new transactions that are made through credit cards that are issued by the company and a core part of the company's operations is to identify all fraudulent transactions in the database.

Proceedings of the VLDB Endowment, Vol. 9, No. 13

To this end, models based on machine-learning techniques have been developed to score each transaction and transactions whose scores are above a certain threshold are classified as fraudulent. However, the models and scoring system are never always precise. There are always some fraudulent transactions that are missed by the models and, likewise, there are always some legitimate transactions that are wrongly identified as fraudulent. For this reason, the credit card companies do not rely entirely on automatically inferred models to determine fraudulent transactions. In addition, they also rely on *rules* that are written by *domain experts*.

Intuitively, a rule captures 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, will capture precisely the fraudulent transactions. The use of rules written by domain experts has the advantage that it allows employing domain knowledge to handle rare special cases. Furthermore, experts can also leverage domain knowledge to detect frauds even before they are fully manifested in the transactions. For example, the credit card company may be informed of potential credit card frauds from a certain IP in Latvia in the coming week. Hence, in this case, the experts can proactively write rules ahead of time to capture those transactions from Latvia. Similarly, an expert may see evidence for fraud purchases in a certain store, and knowing that such frauds often propagate to neighborhood stores, write rules ahead of time to protect surrounding businesses.

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. There is typically already an existing set of rules that were curated by domain experts 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. [1] The goal of RUDOLF, the system that we demonstrate here, is to assist domain experts in achieving the challenging goal.

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 to insert or remove certain items from the query result [3, 2, 8, 10]. However, a key difference here is that such minimal modifications often do not capture the

[1] Likewise, the models that are developed based on machine learning techniques will need to be re-trained.

actual "ground truth", namely the nature of ongoing attack, which may yet be fully reflected in the data. By interacting with domain experts 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.

Finding an optimal set of changes to the rules to capture all fraudulent transactions and omit legitimate transactions is NP-hard in general. In light of this, RUDOLF employs heuristics to assist a domain expert in refining rules. The interactive rule-refinement framework of RUDOLF allows a domain expert to systematically follow-up on the suggestions provided by RUDOLF, fine-tuning them as needed based on her insights and domain knowledge.

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 interact with experts to refine rules. As we shall demonstrate, RUDOLF can also be applied to refine rules to capture evolving network attacks over time (see Section 4). Our experimental results on real-world datasets [5] further prove that RUDOLF is effective and efficient tool for these purposes.

Demonstration Overview We will demonstrate the operation of RUDOLF using two different real-world publicly available datasets. A credit card transactions dataset, which we use to showcase the usefulness of our approach directly for credit card frauds, and a network traffic log, which we use to demonstrate the generality of our solution beyond credit card frauds and its applicability to other fraud scenarios. In both scenarios we will invite VLDB16 attendees to play the role of security experts as well as that of attackers (providing both, in each case, brief relevant background). We will demonstrate how RUDOLF works interactively with the experts to adapt the rules in response the incoming data. In parallel, we will gain an insight into the system operation through its administrator mode screen. See Section 4 for full details.

2. TECHNICAL BACKGROUND

We will briefly present our underlying model, and illustrate main concepts using a simplified example.

Transaction relation A *transaction relation* is a set of tuples (or *transactions*) where, each tuple denotes a transaction. In the context of the credit card scenario, each tuple denotes a purchase made through some credit card. The transaction relation is appended with more transactions over time. We assume that the domain of every attribute A has a partial order associated with it. A transaction may be *fraudulent* which means that the transaction is suspicious and possibly carried out illegally. Conversely, some transactions may be verified to be *legitimate*. Transactions which are neither fraudulent nor legitimate are called *unlabeled* transactions.

Rules Credit card experts specify a set of rules that can be applied to a transaction relation to discover fraudulent transactions. For simplicity and efficiency of execution, the 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. Similarly, for simplicity, our rules contain 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 in this demo. 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 $\alpha_1 \wedge ... \wedge \alpha_n$ where n is the arity of the transaction relation,

Time	Amount	Transaction Type	Location	
18:02	107	Online, no CCV	Online Store	F
18:03	106	Online, no CCV	Online Store	F
18:04	112	Online, with CCV	Online Store	L
19:08	114	Online, no CCV	Online Store	F
19:10	111	Online, with CCV	Online Store	
20:53	46	Offline, without PIN	GAS Station B	F
20:54	48	Offline, without PIN	GAS Station B	F
20:58	44	Offline, without PIN	GAS Station B	F
20:58	47	Offline, with PIN	Supermarket	
20:59	49	Offline, with PIN	GAS Station A	
:	:	:	:	:

Figure 1: Transaction relation on day $n+1$.

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 $\phi(I)$ denotes the set of all tuples in I that satisfies ϕ. We say that $\phi(I)$ are the transactions in I that are *captured* by ϕ. If Φ denotes a set of rules over I, then $\Phi(I) = \bigcup_{\phi \in \Phi} \phi(I)$. In other words, $\Phi(I)$ denotes the union of results of evaluating every rule in Φ over I. Observe that $\Phi(I) \subseteq I$ since every rule selects a subset of transactions from I.

EXAMPLE 1. *To illustrate consider the following simple example. Figure 1 shows a set of transactions, ordered by the time of the transaction, that are recently received. Ignore for now the "F"/"L" labels of some of the transactions. Each tuple also has a score, omitted from the table for conciseness, that shows the degree of confidence (that is derived by a model based on machine learning techniques) that the tuple is fraudulent.*

Consider the following simplified set Φ of existing rules that were used to capture fraudulent transactions until today.

r_{11}) Time $\in$ [18:00, 18:05] $\wedge$ Amt $\in$ [110,115]
r_{12}) Time $\in$ [18:55, 19:00] $\wedge$ Amt $\in$ [110,115]
r_2) Time $\in$ [21:00, 21:15] $\wedge$ Amt $\in$ [40,50] $\wedge$ Location='GAS Station A'

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 around $110-115. The last rule captures a fraud pattern at Gas station A where false charges of amounts between 40 to 50$ are made soon after the closing time at 9pm. In practice each rule also includes some threshold condition (not shown) on the score for each transaction, i.e. the degree of confidence that the transaction is fraudulent, as well as additional conditions on the user/settings/etc. We will omit the scores and the additional conditions for simplicity and focus on the rules in this example. The current set of rules captures only the two shaded tuples shown in the transaction relation. Now, assume that the transactions that are labeled with "F" (resp. "L") are transactions that were reported, e.g. by the card holder, to be fraudulent (resp. legitimate). Clearly, the new fraudulent transactions are not captured by the existing rules and furthermore, the rules wrongly capture some legitimate transaction. The rules thus naturally need to be refined to correctly capture the ongoing fraud attempts and adequately classify all transactions.

The main algorithm behind RUDOLF Our goal is assist the domain experts in refining a set of rules so that, to the extent possible, all and only all fraudulent transactions are captured. In [5], we show that computing an optimal set of modifications to the rules even for special cases of the above problem (when there are new fraudulent transactions and no new legitimate transactions, or the converse: when there are no new fraudulent transactions but only

legitimate transactions) are NP-hard in general. To bypass the hardness results, we develop two heuristic algorithms to identify the best set of rule modifications in these two special cases, then combine them to arrive at a solution for the general problem.

Intuitively, the first algorithm identifies the best modifications to the rules to capture the missed fraudulent transactions. It first clusters all fraudulent transactions (old and new) into groups of similar tuples, and represents each cluster by a generalized representative tuple. The algorithm then proceeds to identify a best rule modifications from a list of top-k rules that will capture the representative tuple (and hence, all tuples) of each cluster. (The top-k rules are the best k rules in terms of minimizing modification costs.) The proposed modifications are verified with the domain expert, who may decline, accept, or further adapt the proposed changes.

As the resulting rules may still wrongly capture some legitimate tuples, our second algorithm attempts to modify the rules to exclude legitimate transactions. For that, the algorithm identifies rules that wrongly capture legitimate tuples and "splits" one of their attributes so that the legitimate tuples are avoided. Intuitively, a split on an attribute A duplicates the original rule but modifies the condition on A so that the legitimate tuple will not be captured. The attribute that is selected is the one that maximizes the "benefit" of the modification. Like before, the expert can choose to decline, accept, or further modify the suggested modifications. As the resulting rules may also exclude some fraudulent transactions, the expert may repeat the two phases until a satisfactory set of rules is arrived.

Full details of the algorithm and our experimental study can be found in [5]. In particular, our experiments compare the fraud prediction accuracy of the rules generated with RUDOLF to that of the common fully-manual approach where experts refine rules themselves, as well as to fully-automatic state of the art Machine Learning detection, demonstrating the superiority or our approach.

EXAMPLE 2. *To continue with our example, note that by slightly adjusting the time and amount intervals and having the third rule cover also Gas station B, we could obtain a set of rules that capture the fraudulent transactions and avoid the legitimate ones. However, such minimal changes often do not correspond to the best or correct changes. By interacting with* RUDOLF*, the domain experts can view/accept/reject/modify the suggestions provided by* RUDOLF*, arriving for instance at the following set of rules.*

r'_1) *Time ∈ [18:00, 19:10] ∧ Amt ∈ [105,115] ∧ Type = Online,no CCV.*
r'_2) *Time ∈ [20:45,21:15] ∧ Amt ∈ [40,50] ∧ Location ≤ Gas Station ∧ Type = Offline, no PIN.*

Intuitively, the updated rule r'_1 conveys the conclusion that the previous first two rules reflect in fact part of a single more general fraud attempt consisting of online transactions without CCV that occur in the evening, charging amounts between $105-115. Similarly, r'_2 reflects the conclusion that the gas station frauds that started at a single station further expand to gas stations in general via offline transactions without code around closing time, of amounts between $40-50.

Each rule above is derived based on refinement suggestions by RUDOLF *and further adaptations by domain experts. For example, rule $r1'$ is obtained because* RUDOLF *determines that a minimal way to change rule R_{11} in Example 1 to capture a good set of fraudulent transactions is to modify the condition "Amt ∈ [110,115]" to "Amt ∈ [106,115]". The domain expert further generalizes the condition to "Amt ∈ [105,115]" since rounded bounds are more common in fraud patterns. So rule R_{11} is modified to*

Time ∈ [18:00,18:05] ∧ Amt ∈ [100,115]

Next, since the first legitimate transaction is captured by this rule, RUDOLF *proposes to exclude it by splitting the rules either on the time attribute (specifically to omit 18:04) into*

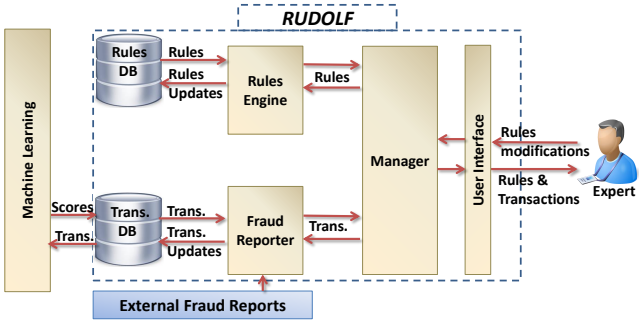

Figure 2: RUDOLF System Architecture

a) *Time ∈ [18:00,18:03] ∧ Amt ∈ [100,115].*
b) *Time = 18:05 ∧ Amt ∈ [100,115].*
or on the Type attribute into
a′) *Time ∈ [18:00,18:05] ∧ Amt ∈ [100,115] ∧ Type ≤ Offline.*
b′) *Time ∈ [18:00,18:05] ∧ Amt ∈ [100,115] ∧ Type = Online,no CCV.*

There is a partial order (not described) that corresponds to the type attribute. Without going into the details of the partial order, the above split on the type attribute will omit transactions that are performed online, with CCV code. Using domain knowledge that only online purchases, especially those without CCVs are of concern, our domain expert chooses the second split option but eliminates the rule (a'). To capture the remaining fraudulent transactions RUDOLF *now proposes to the expert to expand the time interval in (b'), which explains how r'_1 above was arrived (and rule R_{12}, that is now redundant, eliminated). A similar process is carried out for the remaining rules and fraudulent/legitimate transactions, arriving at rule r_2 (and eliminating the now redundant rule R_{12}). Further details on how the final set of rules is derived can be found in our full paper [5].*

3. SYSTEM OVERVIEW

RUDOLF is implemented in PHP (back-end), JavaScript (front-end) and uses MySQL as the database engine. The system architecture is depicted in Figure 2.

Outside RUDOLF, the Machine Learning module scores each transaction and also provides a threshold over which transactions are suspected as fraudulent. These scores may then be used by the rules. The experts connect to, examine the transactions and rules, and adapt the rules through the *User Interface*. Within RUDOLF, there are three core modules: *Manager*, *Rules Engine* and *Fraud Reporter*. The *Manager* module is the main module of the system that executes our algorithm for rules refinement and interacts with all other parts of the system. The *Manager* module interacts with the *Rules Engine* module, which is the module that manipulates (i.e., modifies or add or even delete) the rules that are stored in the *Rules Database*. The *Manager* module also interacts with the *Fraud Reporter* module, which receives external requests to label transactions as fraudulent or legitimate (e.g., when credit card holders notify the credit card company of fraudulent transactions or about legitimate transactions that were wrongly misclassified as fraudulent and thus rejected). Upon receiving reports of fraud/legitimate transactions, the *Fraud Reporter* module passes the transactions that are misclassified by existing rules to the *Manager* module for further inspection by the experts. The expert obtains the list of misclassified transactions and may attempt to modify the existing rules to remedy this. At the same time, the *Manager* also interacts with *Rules Engine* and executes our algorithms to obtain modifications to the existing rules that will help capture all missed fraudulent transactions and avoid misclassifying legitimate transactions. The rules and modifications are displayed via the UI.

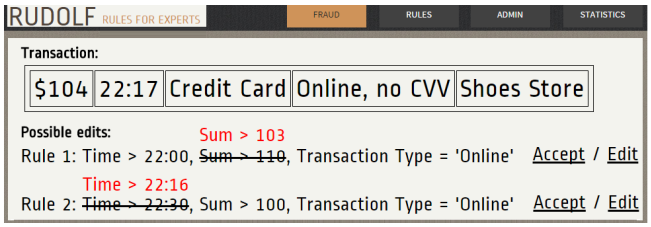

Figure 3: RUDOLF UI: mark incorrect/add missing answers

Figure 4: RUDOLF UI: administrator view

4. DEMONSTRATION SCENARIO

We plan to demonstrate the functionalities of RUDOLF interactively through two scenarios. The first scenario deals with credit cards fraud detection and uses a real-life transactions data set from [4]. This dataset contains attributes including amount, date, card type, location, etc. with some transactions identified as fraudulent and others as legitimate. Transactions from the dataset will be gradually streamed in and the given set of rules will need to be adapted to capture them. The demo attendees will play here two roles: security experts and credit card fraudsters. To the ones playing security experts, RUDOLF will pose questions to solicit feedback on how the existing rules should be modified to capture incoming fraudulent transactions or to write new rules. The users playing the fraudster will be able to introduce new fradulent transactions that form new fraud patterns and we will demonstrate how RUDOLF interactively works with the experts to refine the rules to capture these attacks as well. The second scenario deals with the detection of network attacks. For that we will use network traffic data from network sniffers [9]. Transactions in this dataset contains attributes such as time, source/destination IP, port, etc. Here again we will have the demo attendees play both the roles of security experts as well as network attackers and demonstrate how RUDOLF can interactively modify the rules to capture the new attacks.

To ensure that our demonstration is engaging, we designed an interactive multiplayer game with a scoring system where players that play the experts role are awarded points for writing/modifying rules some that they correctly capture the frauds, with high precision and recall, while attendees playing the fraudsters role are awarded points when their attacks are not fully captures by the derived rules. Since not all the conference attendees are credit card/network security experts, we will provide the audience with information sheets describing some of the the basic principles in each of the domains. We also designed the game to include different levels of challenges, ranging from really simple fraud/network attack patterns to demonstrate the basics of RUDOLF to an advanced fraud/network attacks based on real world scenarios that we obtain by consulting with real domain experts. For example, an attack pattern may be "Packets directed to port 80, of size smaller then 16 bytes". (A minimal HTTP requests consists of 16 bytes, hence this request looks suspicious.) We will begin our demonstration by presenting the game, its goal and rules. We will then allow our audience to play, while we explain the underlying algorithms in RUDOLF at the same time. For example, a player playing an experts role can start the game with a proposed fraud transaction that is not currently captured but needs to be captured by the rules. The first screen presented to the player contains the fraudulent transaction and a list of possible edits to a "best" set of rules in order to capture the fraudulent transaction. The player can either choose an edit from the options, or customize a given edit (Figure 3). Once the fraudulent transactions are captured, the player will continue to the next phase of the algorithms to refine the rules to avoid capturing legitimate transactions (again, by editing existing rules to exclude those transactions). At every stage of the game, the player can opt to see statistics on how many fraudulent transactions are captured, or still omitted, or how many legitimate tuples were caught by mistake. Players that play the fraudsters will use another computer and through a dedicated screen will generate an attack by selecting or providing a fraud pattern. We will leverage the game to elevate and discuss the details and nuances of our algorithms, and especially the decisions taken by RUDOLF that led to the questions it poses. In particular, we will exploit opportunities to explain why certain edits were chosen by the system and the effect of the experts' answers on the system's state. Once players have completed the game, we will display the Administrator web page (Figure 4), which shows the aggregated statistics for all players and attacks (e.g., the total number of rules edited, frauds detected, missing frauds, etc.).

Related work We describe only the main related work next. Traditionally, methods for detecting frauds and intrusions were largely based on machine learning and data mining methods (see, for example, [1, 7]). In fact, a KDD cup was held in 1999 [6] to encourage the development of a network intrusion detector, which is a predictive model capable of distinguishing between "bad" connections, called intrusions or attacks, and "good" normal connections. Our work complements predictive models for intrusion detection. RUDOLF is applied after an initial model for fraud detection is learnt or derived, to capture (resp. clear) additional fraudulent (legitimate) tuples that may be missed (misclassified) by the models and maintain the rules over time. We have mentioned above the relationship to query/rule refinement [3, 2, 8, 10] and the advantage of incorporating experts knowledge in the process.

Acknowledgements This work has been partially funded by the European Research Council under the FP7, ERC grant MoDaS, agreement 291071 and by a grant from the Blavatnik Cyber Security center. Tan is partially supported by NSF grant IIS-1450560 and IIS-1524382.

5. REFERENCES

[1] P. K. Chan and S. J. Stolfo. Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. In *KDD*, pages 164–168, 1998.

[2] A. Chapman and H. V. Jagadish. Why not? In *SIGMOD*, 2009.

[3] S. Chaudhuri. Generalization and a framework for query modification. In *ICDE*, pages 138–145, 1990.

[4] Ucsd data mining contest 2009 (via private communication). `http://www.quansun.com/ucsd-data-mining-contests.htm`.

[5] Rudolf (full version). Submitted. `http://slavanov.com/research/rudolf-full.pdf`.

[6] Kdd cup 99, intrusion detector learning. `https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`.

[7] Y. Kou, C.-T. Lu, S. S, and Y.-P. Huang. Survey of fraud detection techniques. *ICNSC*, 2:749–754, 2004.

[8] D. Mottin, A. Marascu, S. B. Roy, G. Das, T. Palpanas, and Y. Velegrakis. A probabilistic optimization framework for the empty-answer problem. *PVLDB*, 6(14):1762–1773, 2013.

[9] Caida network dataset. http://www.caida.org/data/overview/.

[10] M. Volkovs, F. Chiang, J. Szlichta, and R. J. Miller. Continuous data cleaning. In *ICDE*, pages 244–255, 2014.