Rule Sharing for Fraud Detection via Adaptation (full version)

Ariel Jarovsky[1] Tova Milo[1] Slava Novgorodov[1] Wang-Chiew Tan[2]
[1]*Tel Aviv University* [2]*Recruit Institute of Technology*
[1]{arielhe, milo, slavanov}@post.tau.ac.il [2]wangchiew@recruit.ai

***Abstract*—Writing rules to capture precisely fraudulent transactions is a challenging task where domain experts spend significant effort and time. A key observation is that much of this difficulty originates from the fact that such experts typically work as "lone rangers" or in isolated groups, or work on detecting frauds in one context in isolation from frauds that occur in another context. However, in practice there is a lot of commonality in what different experts are trying to achieve. In this paper, we present the GOLDRUSH system, which facilitates knowledge sharing via effective adaptation of fraud detection rules from one context to another. GOLDRUSH abstracts the possible semantic interpretations of each of the conditions in the rules at the source context and adapts them to the target context. Efficient algorithms are used to identify the most effective rule adaptations w.r.t a given cost-benefit metric. Our extensive set of experiments, based on real-world financial datasets, demonstrate the efficiency and effectiveness of our solution, both in terms of the accuracy of the fraud detection and the actual money saved.**

I. Introduction

Financial frauds are unauthorized financial operations (called transactions) that obtain money or goods illegally from accounts. Financial frauds are a billion dollar industry and financial companies (banks, credit card companies, etc.) invest significant resources to detect frauds and prevent them. Online fraud detection systems monitor incoming transactions and use models based on data mining and machine learning (ML) techniques to detect frauds [17]. A typical approach is to score each transaction and every transaction whose score is above a threshold is classified as fraudulent.

However, such approaches may still not achieve high precision and recall as legitimate transactions may be misclassified as fraudulent and, likewise, fraudulent transactions may be missed. Also, the derived threshold does not provide a semantic explanation of the underlying causes of the frauds like the ways rules do. For this reason, financial companies typically rely, in addition to ML models, on rules that are carefully specified by domain experts. The rules refine the ML scores with domain specific conditions, for automatically determining fraudulent transactions.

Writing rules to capture precisely fraudulent transactions is a challenging task where domain experts spend significant effort and time. A key observation is that much of this difficulty is due to the fact that experts typically work as "lone rangers" or in isolated groups to define the rules, or work on detecting frauds in one context in isolation from frauds in another context. However, in practice there is a lot of commonality in what different experts are trying to achieve. Very often, rules defined by experts in one given context may be useful for understanding how to detect frauds in another context. Such collaboration between experts is desired between the different branches of financial entities, and even between competitors that are willing to share knowledge for the greater cause of battling financial crime (as imposed by local regulations [30]). The goal of the GOLDRUSH system presented here is to facilitate knowledge sharing via effective adaptation of fraud detection rules from one context to another.

We examine the conditions of the rules, and the source and target contexts, to map rules from one context to another. The possible semantic interpretations of each condition are abstracted and then instantiated to the target context. As the number of possible abstractions (and corresponding instantiations) may be large, each rule can be mapped to the target context in many different ways. The efficient algorithms underlying GOLDRUSH identify the most effective ones, in terms of improving the fraud detection accuracy in the target context.

To illustrate, let us consider the following example.

Example 1.1: Consider two fraud detection experts, A and B, working in two different collaborating companies, located in the USA and Germany, respectively. Figures 1 and 2 show a portion of each company's transaction relation (for simplicity, we present here just some real-world attributes, including the transaction time, amount, type, and country). We also present in the last column the ground truth labeling for each transaction: fraud (marked as F) or legitimate (L).

$$\begin{aligned}\varphi^A :& \text{Type} = \text{“}\mathit{Stock\ Trade}\text{”} \wedge \text{Amount} \geq 100K \wedge \\ & \text{Time} \geq \text{16:00} \ \wedge \text{Country} \in \{\mathit{Dinotopia},\ \mathit{Jamonia}\}\end{aligned}$$

The above rule is designed by expert A to detect the frauds in Figure 1. In practice each rule also includes a threshold (not shown) on the transaction score as derived by a Machine Learning module, 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 only on the rules in this example.

The rule defined by expert A in her context may be useful, after some appropriate adaptation, for the context of expert B, and different interpretations will yield different target conditions. Observe that the condition over Time may refer

Time	Amount	Type	Country	
15:58	107K	Stock Trade	Dinotopia	L
16:01	104K	Stock Trade	Dinotopia	F
16:02	111K	Stock Trade	Jamonia	F
16:04	102K	Stock Trade	Dinotopia	F
16:15	96K	Stock Trade	Dinotopia	L
⋮	⋮	⋮	⋮	⋮

Fig. 1: Expert A Transactions

Time	Amount	Type	Country	
19:53	140K	Stock Trade	Orsinia	L
20:02	97K	Stock Trade	Orsinia	F
20:03	230K	Stock Trade	Orsinia	F
20:05	92K	Stock Trade	Orsinia	L
20:07	206K	Stock Trade	Orsinia	F
⋮	⋮	⋮	⋮	⋮

Fig. 2: Expert B Transactions

to the local time or the time after the closing of the local Stock Exchange Market (i.e., 16:00 for the New York Stock Exchange). A time-zone adaptation should be applied for the former interpretation (+6 hours) to adapt it to the time in Germany. However, the target condition should be Time $\geq$ 20:00 (the Frankfurt Stock Exchange closing time) if it is the latter interpretation. Similarly, Country $\in \{Dinotopia, Jamonia\}$ may refer to all over the world attacks originating from these two specific (fictional) countries, in which case an identity mapping should be employed. Conversely, if the condition deals with a specific attack against the local market (USA in this case), we should map it by the rule target country market attackers, e.g. Orsinia. Finally, Amt $\geq 100K$ may be a condition in terms of the local currency, in which case a translation from US dollars to Euro may be applied (translates to about $95K$). Or, it may be a condition that captures "exceptionally large amounts", which, considering the trade amounts distribution in context B, should be mapped to $200K$ Euro. A resulting possible translation for rule φ^A then may be the following:

$$\begin{aligned}\varphi^B : &\text{Type} = \text{"}Stock\ Trade\text{"} \wedge \text{Amount} \geq 95K\ \wedge \\ &\text{Time} \geq 20:00\ \wedge \text{Country} \in \{Orsinia\}\end{aligned}$$

Our goal is to facilitate meaningful and effective rules mappings between different contexts. A fundamental challenge is that rule semantics is often undocumented. To overcome this, we focus on the individual rule conditions and derive a set of candidate value abstraction and concretization functions that may capture the possible mappings between the conditions in the given contexts. These include common built-in mappings (e.g. currency and distribution-based mappings) as well as semantic-aware mappings, derived by analyzing the given data instance. To choose between the possible mappings we define an intuitive cost-benefit model that reflects the improvement in fraud detection that the resulting rules may bring. Finally we provide a set of efficient algorithms to choose the best translation among the candidates set. We show that finding the optimal rule adaptation is NP-hard (intuitively, as all the combinations between the conditions adaptation candidate values need to be considered), but the problem can be modeled as an Integer Linear Programming (ILP) problem. As the performance of ILP solvers often deteriorates for large number of variables (which is the case here), we employ a dedicated data reduction technique that clusters together transactions that "behave uniformly" w.r.t the conditions in the rule, dramatically reducing the number of variables and yielding efficient performance[1]. We further consider the top-k rule adaptation problem and present an effective optimization technique that allows to prune redundant ILP computations.

Our contributions can be summarized as follows.

- We formulate and present a novel rule adaptation framework that employs value abstraction and concretization to map individual rule conditions from one context to another. An intuitive cost-benefit model measures the improvement in fraud detection that the resulting rules may bring.
- We show that the problem of identifying the best rule adaptation is NP-hard and propose an effective data reduction technique which, together with a dedicated ILP formulation of the reduced problem, yields nevertheless a practically efficient performance.
- We further consider the generalized problem of computing, given a set of rules, the top-k best adaptations and present an optimization technique which computes an upper bound on the potential cost-benefit contribution of a given rule, and then uses it to successfully prune redundant ILP computations.
- We have implemented our solution in the GOLDRUSH prototype system and applied it on real use cases, demonstrating the effectiveness and efficiency of our approach as a whole, as well as the individual contribution of each components. We performed experimental evaluations on five real-life datasets of financial transactions that showed how our algorithms consistently outperformed alternative baseline solutions, both in terms of the accuracy of the frauds detection and the actual money saved.

We note that while some previous work has studied the problem of adapting rules to changes in the data (e.g. [31], [13], [18]), they mainly focus on incremental maintenance of rules in a *given context* as new transactions come in. In contrast our work examines rule adaptations entailed by context switching. Closest to our work is research on transfer learning, that attempt to reuse knowledge gained while training a model in one context to save training cost in another context. In particular fraud detection rules may be viewed as a form of decision trees to be transferred between contexts. However, since their objective function is different, as we show in our experiments, not only the resulting decision trees after the ML-based transfer are less meaningful in terms of (semantics)

[1]While the ILP solver still goes through exponentially many combinations, we have found that it gives very good performance in practice on the reduced data.

understability to the human expert, they are also less effective in terms of fraud detection. We further discuss related work in Section VI.

While we focus here on a particular application domain, our techniques are generic and may be generally utilized to adapt classification rules under similar cost-benefit objectives.

The paper is organized as follows. Section II presents the data and cost-benefit model that we employ and formalizes the rule adaptation problem. The problem complexity as well as the optimized algorithms to solve it are considered in Section III. The classes of value mappings that we consider are detailed in Section IV. The experiments are described in Section V. Related work is presented in Section VI, and we conclude in Section VII.

II. Preliminaries

In the following section we present all the necessary definitions to formally define the problem. We start by explaining the data model (transactions and rules) followed by the definition of rule adaptation and the cost/benefit model that we employ.

A. Transactions

Our model consists of a *transaction relation* $T(A_1, A_2, \ldots, A_m) = \{t_1, \ldots, t_n\}$, which is a set of tuples (or *transactions*) over the attributes $A_1, A_2, \ldots, A_m$. In a financial context, a tuple t_i captures an operation (i.e. transfer, payment) made through some bank or credit card. The transaction relation is appended with more transactions over time. For brevity, when the attributes and the transactions set are clear, we will omit them and use just T. Every attribute A_i belongs to a domain $dom(A_i)$ which has a partial order that is reflexive, antisymmetric and transitive, with a greatest element $\top_A$ and a least element $\bot_A$. Attributes associated with a partial order but not with a total order are called *categorical* attributes. The elements in such partial order are sometimes referred to as *concepts*. W.l.o.g. we also assume that $\top_A$ and $\bot_A$ do not appear in any of the tuples [2]. These special elements will be useful when defining rule mappings. For brevity, when an attribute name is clear from the context we omit it and simply use the notations $\top$ and $\bot$.

A transaction may be classified *fraudulent* which means that the transaction was carried out illegally. Conversely, a transaction may be verified to be *legitimate* and classified accordingly. The labeling is assumed to correspond to the (known part of the) *ground truth*. Given a transaction relation T we denote the set of fraudulent (legitimate resp.) transactions as $fraud(T)$ ($legit(T)$). In addition, each transaction is attached with a score between 0 and 1, (computed automatically using machine learning techniques) which represents the estimated probability of each transaction to be fraud. Figures 1 and 2 depict two sample transaction relations. For the simplicity of the presentation, the score is omitted from the figures. Also, we assume (for simplicity) that all contexts have the same schema. Otherwise one may apply schema matching techniques before employing our algorithms.

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

B. Rules

Fraud detection experts specify rules that can be applied to a transaction relation to identify the fraudulent tuples. For efficiency of execution, the rules are typically written over a single relation, which is a universal transaction relation that includes all 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. A *rule* φ in our setting is defined as a conjunction of one or more conditions over the attributes of a transaction relation T, 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. Note that the rule language that we consider, albeit simple, forms the core of common rule languages used by actual systems and real-world industry companies [18]. Indeed, shown in the experiments, it covered all the obtained rules.

A *rule* φ is defined as a conjunction of one or more conditions over the attributes of a transaction relation T, i.e. it is of the form $\varphi = \bigwedge_{1 \le i \le m} \alpha_i$ where α_i is a condition of the form '$A_i\ op_i\ v_i$', $op_i \in \{=, \neq, <, >, \le, \ge, \in, \notin\}$. We assume that the rules are well-defined: the v_i values belong to the corresponding attribute's domain and the operators' semantics is the corresponding from the domain. For readability, in our examples we show only the non-trivial conditions on attributes, namely omit conditions of the form $A_i \le \top$. 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. 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 [18].

The semantics of a rule φ is defined as a boolean function over the tuples of a relation T such that for a tuple $t = < A_1 : a_1, A_2 : a_2, \ldots, A_m : a_m >$,

$$\varphi(t) = \bigwedge_{1 \le i \le m} \alpha_i[a_i/v_i]$$

Here a_i/v_i denotes the replacement of v_i by a_i in α_i, and the semantics of $\alpha_i[a_i/v_i]$ is defined by the standard op_i semantics, except for the case where $a_i = \top$ for which α_i is always satisfied. Similarly, the semantics of applying a rule φ on T denote the set of all tuples that are *captured* by φ, i.e.: $\varphi(T) = \{\ t \in T \mid \varphi(t) = True\}$. Let Φ denote a set of rules over T, then $\Phi(T) = \bigcup_{\varphi \in \Phi} \varphi(T)$. In other words, $\Phi(T)$ denotes the result of the union of evaluating every rule in Φ over T. Observe that for every T and for every $\varphi \in \Phi$, $\varphi(T) \subseteq \Phi(T) \subseteq T$, since every rule selects a subset of transactions from T.

As an example, the rules φ^A and φ^B from the Introduction are rules in the presented language.

C. Rule-Driven Transactions Classification

Recall that transactions may be associated with a classification label that indicates whether they are fraudulent or legitimate. Given a rule φ and a set of transactions T (typically the full transaction relation), we define the set of *fraudulent transactions captured by* φ as $F_C(\varphi, T) = fraud(T) \cap \varphi(T)$. Conversely, the set of *uncaptured fraudulent* transactions is defined as $F_U(\varphi, T) = fraud(T) \setminus F_C(\varphi, T)$. The set $L_C(\varphi, T)$ of *legitimate transactions (wrongly) captured by* φ and the set $L_U(\varphi, T)$ of *uncaptured legitimate* transactions are similarly defined, by replacing $fraud(T)$ with $legit(T)$ (and correspondingly replacing F_C with L_C in the L_U formula).

Notice that these definitions can be extended also to be used with a rule set Φ by simply replacing φ with Φ. For brevity, when the rule/rules set and the transaction relation are clear, we will omit them and simply use the notations F_C, F_U, L_C and L_U. Ideally, we aim to build a set of fraud detection rules in which F_C and L_U are as large as possible (and so F_U, L_C are as small as possible).

D. Rule Adaptation

The *context* of an expert includes her transaction relation and her current set of fraud-detection rules, as well as relevant contextual information such as the company she works for, its location, currency, language, local regulations, etc.

While there might be many ways to map rules from a source context to a target, we center our attention here on mappings that consider the individual conditions of the rule, substituting (when needed) the values v_i used in the rule conjuncts by those that best match the target context. As we will see in Section V, such value-based mappings are extremely effective.

Which value substitutions should one consider when adapting a rule from one context to another? As illustrated in our running example from the Introduction, there are multiple semantics to consider for each condition (attribute). For instance, the value in the *Amount* attribute conjunct can represent an absolute amount in the local context currency or some regulatory value, or alternatively correspond to some percentile in the amounts distribution. The candidate mappings are thus obtained by first abstracting the value from the source context, using each of these possible relevant semantics, and then concretizing the abstracted values to the target context.

More formally, given an attribute A_i, we denote by $\Sigma(A_i)$ its given set of possible semantics. For a context s, an attribute A_i and a possible value semantics $\sigma \in \Sigma(A_i)$, the *attribute abstraction function* $\alpha^s_\sigma : dom(A_i) \to dom(\sigma)$ maps values from the attribute domain $dom(A_i)$ to the abstract semantic domain $dom(\sigma)$. Conversely, the *attribute concretization function*, γ^s_σ, is the inverse function that maps an abstract semantic value $y \in dom(\sigma)$ to its set of possible origins in $dom(A_i)$, except for the special $\top$ element which is mapped to itself: $\gamma^s_\sigma(y) = \{x | \alpha^s_\sigma(x) = y \text{ if } y \neq \top, \text{ else } \top\}$

To map a rule φ from a source context s to a target context t, we will consider, for each conjunct $A_i\ op\ v_i$ in φ, its possible abstractions (under the different semantic $\Sigma(A_i)$), and then concretize each to the target domain. Formally,

Semantics	**Abstraction**	**Concretization**
Identity [**ID**]	100K	100K
Exchange Rate [**CC**]	*97K (CHF)*	95K(EUR)
Distribution [**VP**]	*upper 5%*	200K
Regulation Limits [**RL**]	*after hours*	120K

Fig. 3: Mapping of $\{Amount \geq 100K\}$ from context A to B

Definition 2.1 (Attribute Mapping Candidates): Given a source context s, a target context t and an attribute A_i, the *mapping candidates* for a value $v_i \in dom(A_i)$ are the values

$$V_i^{st}(v_i) = \bigcup_{\sigma \in \Sigma(A_i)} \gamma^t_\sigma(\alpha^s_\sigma(v_i))$$

When s, t and v_i are clear from the context, we will omit them and use just V_i.

Example 2.2: To continue with our example from the Introduction, consider the two financial institutes A and B that want to share the knowledge about fraud attacks. Let us focus on the Amount attribute of the first rule φ^A. Naturally, there are several possible mappings for this attribute. For example one of them may use the exchange rate of US Dollars (the local currency of A in USA) to Euros (the local currency of B in Germany). In this case the abstraction here maps the local amount to some agreed upon reference currency (Swiss Francs in the example); a conversion from the reference currency to Euro is then used as the concretization function (marked as **CC** in Figure 3). A second possible option is a distribution-based mapping that maps the value to its corresponding percentile (e.g. upper 5%) among the transaction amounts in context A. The concretization function then maps the abstract percentile to the corresponding concrete value in the target context B (marked as **VP**). Another possibility is using semantics that maps the amount to some local financial regulation, assuming that the value of the attribute in the rule is exactly the regulation limit. Here the abstraction will be the "type of regulation" (e.g. after hours trade) and the concretization is the value of the corresponding regulation limit in the target context country (marked as **RL**). In addition, there is an identity mapping that leaves the value unchanged (marked as **ID**). The summary of the different mappings discussed is depicted in Figure 3.

We are now ready to define the set of candidate rule mappings. Intuitively, it includes all the rules obtained by substituting each of the values appearing in φ by one of its corresponding possible mappings.

Definition 2.3 (Rule Mapping Candidates): Given a source context s with a rule $\varphi = \bigwedge_{1 \leq i \leq m} A_i\ op_i\ v_i$, and a target context t. Let V_i $(1 \leq i \leq m)$, be the set of attribute mapping candidates for v_i. The set of rule mapping candidates for φ is the set of rules

$$\Psi^{st}(\varphi) = \{\varphi[v'_1/v_1, \ldots, v'_m/v_m] \mid v'_1 \in V_1, \ldots, v'_m \in V_m\}$$

Here again, when s and t are clear from the context we omit them and simply use $\Psi(\varphi)$.

E. Cost & Benefit model

As previously explained, to compare between the different rule candidates in $\Psi(\varphi)$ and determine which is the most suitable mapping, we need to measure the "cost & benefit" it entails. Intuitively, the gain from a new rule can be measured by the increase in the number of fraudulent transactions that are captured by adding it (i.e. the fraudulent transactions that were not captured by the existing rules), minus the number of legitimate transactions that it misclassifies (i.e. legitimate transactions that were correctly classified by previous rules).

Problem 2.4 (Best Rule Adaptation Problem): Let s be a source context with a rule φ. Let t be a target context with transaction relation T_t and an existing set of rules Φ_t. Let $F_C,\ F_U,\ L_C$ and L_U denote the set of captured and uncaptured fraudulent and legitimate transactions in t w.r.t to T_t and Φ_t (as defined above). Let $\Psi(\varphi)$ be the rules mapping candidates set as previously defined.

The BEST RULE ADAPTATION PROBLEM is to compute a rule adaptation $\varphi' \in \Psi(\varphi)$ such that:

$$\begin{aligned} w(\varphi') =& (\alpha \cdot |\varphi'(F_C)| + \beta \cdot |\varphi'(F_U)|) - \\ & (\gamma \cdot |\varphi'(L_C)| + \delta \cdot |\varphi'(L_U)|) \end{aligned} \tag{1}$$

is maximized for a given $\alpha, \beta, \gamma, \delta \geq 0$.

The term $(\alpha \cdot |\varphi'(F_C)| + \beta \cdot |\varphi'(F_U)|)$ represents the benefit that can be obtained from adding the rule adaptation φ' to the existing rule set Ψ_t in terms of the fraudulent transactions captured, while the term $(\gamma \cdot |\varphi'(L_C)| + \delta \cdot |\varphi'(L_U)|)$ represents the rule costs (in terms of legitimate transactions captured). By weighting each of the components one can tune, depending on the application, the precision and recall of the selected rule. For instance, setting $\alpha = 1,\ \beta = 1,\ \gamma = 0,\ \delta = 0$ will maximize the recall of the new rule set, while the dual assignment ($\alpha = 0,\ \beta = 0,\ \gamma = 1,\ \delta = 1$) will maximize its precision. A more balanced function (which represents the common policy of financial companies) can be $\alpha = 0.5,\ \beta = 1,\ \gamma = 0.5,\ \delta = 1$. This function aims to capture as much uncaptured fraudulent transactions as possible (especially new ones), while at the same time maintaining a low number of (new) false positives.

Additional constraints (e.g. thresholds on the precision/recall/number of captured legitimate transactions) may be added to the problem definition, to fine tune the adaptation. For simplicity we omit this here but our algorithms extend to such more refined definition.

III. Finding The Best Rule Adaptation

Let us first assume that we have a single rule that we wish to adapt. We will discuss the case where multiple rules are available afterwards. From the exposition in the previous section it follows that there are two main challenges to overcome to yield a good rule adaptation: (1) the choice of suitable abstraction/concretization functions that will allow to build an effective yet not too large set of candidate mappings for each rule attribute, and (2) the design of efficient algorithms to identify the best, cost&benefit-wise, rule adaptation. We will first discuss (2) in this section, assuming that the set of possible mappings for every attribute value is given. Then, in the next section, we will explain which mappings are used.

A naïve algorithm to identify the best rule adaptation would iterate over all possible attribute mappings, evaluate Equation 1 for each of the mappings, and choose the one with maximal value. However, this algorithm's complexity is exponential in the number of attributes and becomes prohibitively expensive when the relations contains many attributes, each with multiple possible mappings, as it is often the case in practice (see Section V). Indeed, we can show the problem to be NP-hard.

Theorem 3.1: Testing whether a rule φ has an adaptation $\varphi' \in \Psi(\varphi)$ whose score $w(\varphi')$ exceeds a given threshold θ is NP-Hard in the number of attributes in the transaction relation, even if each attribute has only two possible mappings.

The proof is presented in Appendix A.

However, as we will show below, the problem can be modeled as an Integer Linear Programming (ILP) problem. While this is still NP-hard, ILP solvers are known to be efficient in practice especially if the number of variables is not too large. Unfortunately, in our setting, the variables in the ILP formulation correspond to the number of transactions, which are typically in the order of millions of them. Hence, we employ a dedicated preprocessing data-reduction step that clusters together transactions that behave uniformly w.r.t the conditions in the rule, representing them as a single tuple, and thereby significantly reducing the ILP problem space. Our experimental results (shown in Section V) on real-world datasets prove the efficiency of our solution.

We will start our exposition by describing the direct ILP formulation of the problem, then explain how it is optimized via data reduction. Finally, we will consider the general case which, given a set of candidate rules, computes a set of top-k rule adaptations.

A. ILP Formulation of the Best Rule Adaptation Problem

Consider a rule $\varphi = \bigwedge_{1 \leq i \leq m} A_i\ op_i\ v_i$, and let V_i, $i = 1 \ldots m$, be the set of attribute mapping candidates for v_i. Intuitively, our ILP problem consists of a boolean variable for every value in each V_i (with the value 1 symbolizing that the corresponding mapping was chosen, and 0 that it was not). We also have a boolean variable for each of the tuples in the target transaction relation T_t (with the value 1 symbolizing that it satisfies the rule under the chosen mapping). The objective function will be analogous to the one defined in problem 2.4. We also have a constraint for every attribute A_i in the rule, ensuring that a single value is selected from V_i. Formally, we define the ILP as follows:

Notations Let $\pi(\Gamma)$ be the set of indexes of transactions in T_t belonging to the set Γ for $\Gamma \in \{F_C,\ F_U,\ L_C,\ L_U\}$. Let $c_i = |V_i|$ be the number of different mapping candidates for attribute A_i and let $n = |T_t|$, then:

Boolean (0-1) Variables:

$x_i = 1$ iff transaction t_i was captured by the chosen rule adaptation

$A_{ij} = 1$ iff mapping candidate j of the set V_i was selected for attribute A_i

Constants:

$\alpha_{ijk} = 1$ if attribute A_i on transaction t_k is satisfied by mapping A_{ij}, otherwise 0

Model Formulation Our goal is to maximize the following objective function under the given set of constraints.

Objective Function:

$$\sum_{i \in \pi(F_C)} \alpha \circ x_i + \sum_{i \in \pi(F_U)} \beta \circ x_i - \sum_{i \in \pi(L_C)} \gamma \circ x_i - \sum_{i \in \pi(L_U)} \delta \circ x_i$$

Constraints:

First, we want to allow just one mapping to be selected for every attribute i:

$$\underset{1 \le i \le m}{\forall} \sum_{j=1}^{c_i} A_{ij} = 1 \tag{2}$$

Then, we have the constraints that define if transaction k is satisfied by the selected adaptation: we require for every attribute A_i that if j was the selected mapping among the candidates in V_i, then the constant α_{ijk} must be equal to 1. For this purpose, we check the transaction satisfaction by counting the number of attributes satisfied by the chosen candidates. Thus, a count result of m is equivalent to a full record satisfaction. We use a lower bound equation and an upper bound one in order to force the variables x_k to behave as expected.

Upper bound: On the first hand, we want x_k to be 1 only in case of full satisfaction. Thus, we need an expression that will allow x_k to be 1 only if there are at least m attributes satisfied.

$$\underset{1 \le k \le n}{\forall} m \cdot x_k \le \sum_{i=1}^{m} \sum_{j=1}^{c_i} \alpha_{ijk} \cdot A_{ij} \tag{3}$$

Lower bound: On the other hand, we want x_k to be 1 in every case of full satisfaction. Thus, we need an expression that will require x_k to be at least 1 if there are m attributes satisfied.

$$\underset{1 \le k \le n}{\forall} m - 1 + x_k \ge \sum_{i=1}^{m} \sum_{j=1}^{c_i} \alpha_{ijk} \cdot A_{ij} \tag{4}$$

This concludes the construction.

The size of the ILP problem, as constructed above, is quadratic in the number of transactions in the target relation T_t. More precisely, it includes $3n + m + \Sigma|V_i|$ constraints over $n + \sum |V_i|$ variables, where n, m are the number of transactions and attributes in T_t, resp. As will be shown experimentally in Section V, this presents a performance bottleneck when the number of transactions in T_t is large. To overcome this, we use a data reduction technique that allows to represent a set of "indistinguishable" transactions by a single tuple.

B. Optimization via Data Reduction

Given a rule φ, its sets V_i, $i = 1 \ldots m$, of attribute mapping candidates, and the target relations T_t, we build a reduced relation T'_t which we call the *RMC (Rule Mappings Clustered) relation*. Then we define a reduced ILP problem for the compressed relation.

The reduced RMC relation: The RMC relation is built by clustering together sets of transactions that would be equally labeled by the adapted rule, independently of which specific attribute mappings are chosen.

Example 3.2: To illustrate, recall our running example from the Introduction. Consider expert B's transaction relation from Figure 2, and rule φ^A that we wish to adapt to her context. Assume that expert B has no other rules at the moment and so, all the fraudulent and legitimate transactions in the figure are currently uncaptured. Suppose that the attribute mapping candidate for adapting φ^A to context B are $V_{Amount} = \{95K, 100K, 120K, 200K\}$ and $V_{Time} = \{16{:}00, 20{:}00\}$. Consider the third and fifth transactions in the Figure. One can verify that, independently of the chosen mappings, both transactions will be both captured or uncaptured by the chosen rule adaptation (since both $Amount$s are above $200K$ and both $Time$s are above 20:00). Our RMC construction, described below, merges such indistinguishable tuples.

Intuitively, for every attribute A_i, we partition its domain $dom(A_i)$, using the values in V_i. We choose (as will be explained below) a representative for each partition, then replace each attribute value in the transaction relation by the representative of the partition to which it belongs. Recall that conjuncts in the rule are of the form $A_i \; op_i \; v_i$. The domain partitioning (and chosen representatives), and consequently the value replacements, are dictated by operator op_i used for value comparison. We use below $h_i^{op_i}$ to denote the value replacement function for attribute A_i.

To illustrate, let us consider the operators $\ge$ and $=$ (the other operators work similarly). For $\ge$, when the domain is totally ordered, we use the values in V_i to partition the domain into disjoint intervals $[v, v')$, $v, v' \in V_i \cup \{-\infty, \infty\}$, and set each partition representative value as the minimal value in the partition (v). (For the case of $v = -\infty$ we use the special $\perp$ symbol). The value replacement functions $h_i^{\ge}$ for attribute A_i is then defined as follows:

$$h_i^{\ge}(v) = argmax_{b \in V_i} \{v \ge b\}$$

For $=$, each element in V_i forms a partition of itself whereas all the others values belong to a "complement" partition whose representative is the $\perp$ element. The value replacement functions $h_i^{=}$ for attribute A_i is then defined as follows:

$$h_i^{=}(v) = v \text{ if } v \in V_i \text{ otherwise } \perp$$

Example 3.3: Following Example 3.2, the next table presents for each $Amount$ value in Figure 2, its partition and representative as induced by V_{Amount} and $h_{Amt}^{\ge}$.

Time	Amount	F_C	F_U	L_C	L_U
16:00	120K	0	0	0	1
20:00	$\perp$	0	0	0	1
20:00	95K	0	1	0	0
20:00	200K	0	2	0	0

Fig. 4: Expert B RMC Transactions Relation

v_i	$v_i's$ *partition*	$h^{\geq}_{Amt}(v_i)$
$92K$	$(-\infty, 95K)$	$\perp$
$97K$	$[95K, 100K)$	$95K$
$140K$	$[120K, 200K)$	$120K$
$206K$	$[200K, \infty)$	$200K$
$230K$	$[200K, \infty)$	$200K$

Suppose also that V_{Type} = {Stock Trade, Payment}. Clearly, using $h^{=}_{Type}$, all the transactions in Figure 2 will remain with the *Stock Trade* type.

Finally, we cluster indistinguishable tuples and attach, to each representative tuple, counters for the four transaction classes
(L_C, L_U, F_C, F_U), counting the number of tuples in the cluster belonging to the corresponding class.

Example 3.4: Continuing with Example 3.2, Figure 4 shows the *RMC table* for the 5 transactions in Figure 2 (we assume that all of them are still uncaptured). We omit the *Type* and *Country* attributes since they have the same values for all the records in the example (and thus will be indistinguishable for any possible adaptation).

The reduced ILP problem: We can now define, for the reduced RMC table, a corresponding reduced ILP Model. In this model, each of the x_i variables represents a summarized transaction in the RMC relation, and the α_{ijk} constants are defined w.r.t the x_i summarized tuples. The definition of the A_{ij} variables stays the same.

We use the same set of constraints as before and adapt just the objective function. Let χ_i^{Γ} denote the number of original transactions of class $\Gamma \in \{F_C, F_U, L_C, L_U\}$ that were summarized into the summarized tuple x_i, then the updated objective function becomes:

$$\sum_{i \in \pi(F_C)} \alpha \circ \chi_i^{F_C} \circ x_i + \sum_{i \in \pi(F_U)} \beta \circ \chi_i^{F_U} \circ x_i -$$
$$\sum_{i \in \pi(L_C)} \gamma \circ \chi_i^{L_C} \circ x_i - \sum_{i \in \pi(L_U)} \delta \circ \chi_i^{L_U} \circ x_i$$

As demonstrated in the experiments, the smaller size of the transaction relation, and consequently the smaller size of the ILP model, leads to significant performance improvement.

C. k-Rule Adaptation

So far we have explained how the best rule adaptation, for a single rule φ is computed. When a set of n rules is available (from the same or different sources), the same algorithm can be applied to each of the rules φ in the set (i.e. computing a reduced RMC relation for φ and running ILP on it). The scores of the best adaptation for each of the rules are compared, and the one with the highest score may be selected and added to the target rules set. The process may then be iterated, to choose another rule adaptation, and so on. We refer below to the available set of candidate rules as the *rules pool*. Let n be the number of rules in the pool and let k the number of iterations (selected rule adaptations) performed. The iterative process just described takes time $n \times k \times I$, where I is the time it takes to compute the relevant RMC and solve its (reduced) ILP problem. Interestingly, however, as we show next, much of this work can be saved by avoiding useless RMC and ILP computations, namely ones having no potential to yield a best-score adaptation. We explain this next.

Consider an iteration $i \geq 1$ and let $\varphi_1, \ldots, \varphi_{i-1}$ be the rule adaptations selected in the previous iterations. Let $\Phi_i = \Phi_t \cup \{\varphi_1, \ldots, \varphi_{i-1}\}$ be the target set of rules at the i^{th} iteration, where Φ_t is the existing set of rules at the target context. Correspondingly, let F_C^i, F_U^i, L_C^i, L_U^i denote the fraudulent/legitimate (un)classified transactions as induced by Φ_i. Note that the update operation of the sets from round $i-1$ to round i can be computed in a fast way by updating just the labels of the transactions captured by φ_{i-1}).

By this way, rule adaptations equivalent to the previous selected have now a different target score (mostly much lower in practical target functions) and new adaptations will be selected.

The proposition presents the key insight of our pruning technique.

Proposition 3.5: Let φ be a rule in the rules pool. Let $w_i(\varphi)$ be φ's best adaptation score as computed at the i^{th} iteration. Let φ_{i-1} be the rule adaptation selected at round $i-1$. Then, $w_i(\varphi)$ bounded by:

$$\widehat{w}_i(\varphi) = w_{i-1}(\varphi) + \alpha \circ |\varphi_{i-1}(F_U^{i-1})| + \delta \circ |\varphi_{i-1}(L_U^{i-1})|$$

where α, δ are the constants defined in the target score formula in Problem 2.4.

The proof is presented in Appendix A.

Corollary 3.6: Let φ be a rule in the rules pool, and let $w_i, \widehat{w}_i$ be defined as in Proposition 3.5. The ILP computation for φ may be skipped at iteration i if there exists another rule φ' in the candidate rules set, for which $w_i(\varphi') \geq \widehat{w}_i(\varphi)$.

Algorithm 1 presents the *k-RuleAdaptation* algorithm which utilizes the above Corollary to prune redundant (RMC and) ILP computations. The rules are stored in an ordered data structure (e.g., a max-heap) that maintains for each rule its current maximal adaptation score (we assume that it has also an *increaseKeys* function that allows to increase all its keys by a given factor at the same time). Then, at each iteration, we determine which ILP computations need to be performed and which will be skipped.

The algorithm takes as input the transaction relation of the target context T_t (with it's initial transactions classification $F_C^1, F_U^1, L_C^1, L_U^1$), its current set of rules, Φ_t, the desired number k of new rules, and threshold θ on the adapted rules score (which can be used to prune rule adaptations with low score). Line 1 initializes the initial rule set, Φ_1, with the existing rule set Φ_t and line 2 initializes the ordered rules pool (e.g., heap), whose keys will be the best rule adaptation score obtained at the current round (w_i), or its upper bound estimator ($\widehat{w}_i$). Each rule φ will also contain an attribute which indicates the last round in which the rule's best adaptation was last called for it ($\varphi.round$, initialized to $null$). Line 3 contains the k-rounds *for* loop, and line 4 contains the *while* loop which looks for the best rule adaptation among all the rules (w.r.t the current transactions classification). Line 5 pops from the heap the rule adaptation with the highest score into φ_i. Lines 6-7 check if the score went below the passed threshold, and if so the algorithm ends and returns the list of rule adaptations found up to this point. Lines 9-13 handle the case in which φ_i's best rule adaption was calculated in the current round. In this case we rely on Corollary 3.6 to deduce that all the other rules in the pool: (i) were evaluated in this round, with score lower than φ_i, or (ii) it's estimator is below $\varphi.score$. At this step, we increase all the keys in the data structure (to update the rule score to the estimator $\widehat{w}_{\text{i+1}}$[3]) through the function *increaseKeys*, add φ_i to the new set of rules, $\Phi_{\text{i+1}}$, and calculate the labeled classification sets $F_C^{\text{i+1}}, F_U^{\text{i+1}}, L_C^{\text{i+1}}, L_U^{\text{i+1}}$ (as induced by $\Phi_{\text{i+1}}$). Finally, lines 15-17 handle the case in which φ_i's best adaptation was not calculated in the current round, then it calculates it's best adaptation w.r.t the current T_t's classification labels and pushes φ_i back to the heap, this time with the updated score. The effectiveness of the algorithm is presented in Section V.

Algorithm 1: k-RuleAdaptation($T_t,\ \Phi_t,\ k,\ \theta$)

```
1  $\Phi_1 \leftarrow \Phi_t$
2  $h \leftarrow heap(RulesPool, \infty)$
3  for $i \leftarrow 1, \ldots, k$ do
4      while true do
5          $\varphi_i \leftarrow$ popMax($h$)
6          if $\varphi_i.score < \theta$ then
7              return $\Phi_i$
8          if $\varphi_i.round == i$ then
9              $\Phi_{i+1} \leftarrow \Phi_i \cup \{\varphi_i\}$
10             increaseKeys($h$,
                 $\alpha \circ |\varphi_i(F_U^i)| + \delta \circ |\varphi_i(L_U^i)|$)
11             $F_C^{\text{i+1}}, F_U^{\text{i+1}}, L_C^{\text{i+1}}, L_U^{\text{i+1}} \leftarrow$ calculateLabels($T_t$,
                 $\Phi_{\text{i+1}}$)
12             updateLabels($T$, $\varphi$)
13             break
14         else
15             $\varphi_i.round \leftarrow i$
16             findBestRuleAdaptation($\varphi_i$, $T_t$)
17             push($h$, $\varphi_i$, $\varphi_i.score$)
18 return $\Phi_{k+1}$
```

[3]Note that the estimator used in the algorithm might be calculated using the previous round estimator ($\widehat{w}_i$). Even though, it is still a proper estimator since it is an upper bound of the estimator presented in Proposition 3.5

IV. Generating Mapping Candidates

We employ in GOLDRUSH three classes of abstract-based attribute mappings that may be used. The first class consists of a standard build in set of value-based mappings. The second includes distribution based mappings. The third class is data driven and employs an ontological knowledge to determine possible abstractions/concretizations. We briefly overview each class below.

A. Mapping by Value

Our set of value-based mappings includes standard value conversions for currency, temperature, time zone, length and weight metrics. The corresponding abstraction (resp. concretization) functions here map, in each context, every value from (to) its local scale to (from) a universally agreed one.

We also include in this class three useful mappings:

Identity: This is the trivial mapping that leaves the values unchanged. Here both the abstraction and the concretization functions are simply the identity function.

Any-to-Any: This mapping allows to replace a value by any other value in the domain. Such a mapping is useful for semantic-less categorical attributes such as category code names or internal user ids, where the corresponding value in the target domain is unknown at mapping time, and allows the algorithm to examine all possible instantiations. The abstraction function here maps all values to the abstract "Any" element. The inverse concretization then facilitates all possible mappings.

Wildcard: This mapping allows to replace a value by the special $\top$ element, which satisfies all conditions, thereby essentially removes the corresponding conjunct from the rule. This is particularly useful when a given rule has conditions that are relevant to the source context but not to the target. For instance conditions on the state name may be relevant to US bank branches but not for the European ones).

B. Mapping by Distribution

The second class includes distribution-based mappings such as percentile and frequency (top/bottom-k). We distinguish here between numerical and categorical attributes.

Numerical attributes: For numerical attributes there is often a correlation between the value used in the condition and the attribute's value distribution in the underlying relation. For example, assuming that only 5% of the money amounts recorded in a transaction relation are above $100K$, a condition of the form of $\{Amount \geq 100K\}$ may in fact mean *"in the top 5% amounts"*. To capture this we use an abstraction function that maps each value to its corresponding percentile in the attribute values distribution, in the source relation. The inverse concretization function then uses the target's distribution to map back to the appropriate target value. (A *bottom n%* mapping works in a similar manner.)

In our implementation we examined two standard methods for estimating the percentiles and their values: building a *Cumulative Histogram* and using *1-D Kernel Density Estimations* (*KDE*) [27]. As both methods yielded almost the same results we show results only for Cumulative Histograms.

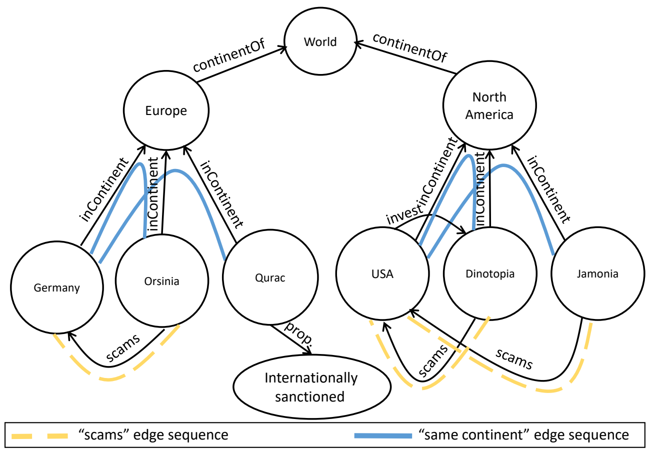


Fig. 5: A sample of geopolitical ontology

Categorical attributes: The *top-k (resp. bottom-k) most frequent* is the analogous version of the numerical *top n% (bottom n%)*, for categorical attributes. The abstraction function here maps every attribute value to the corresponding maximal (minimal) k. The inverse concretization function then maps each k to the appropriate value in the target context.

C. Mapping by Semantics

For categorical attributes, we use ontologies to examine the semantic relationship between the values mentioned in the attribute condition (e.g. Dinotopia and Jamonia in $country \in \{Dinotopia, Jamonia\}$) and each of the values mentioned in the source context attributes (e.g. USA in $location = USA$ of expert A in the Introduction). We use in GOLDRUSH general purpose ontologies such as YAGO [28] and DBPedia [11] as well as domain specific ontologies built by our collaborating domain experts (details in the experiments section).

Specifically, the simple, yet effective, semantic inference that we employ examines the ontology graph. It identifies, for each semantically meaningful value mentioned in the source context, simple path expressions (sequence of labeled edges) connecting the value to the values mentioned in the condition. Each such path expression captures a possible semantic abstraction. To illustrate, two possible path expressions connect the USA node to the two nodes labeled Dinotopia and Jamonia: (1) a single (yellow) edge $scams$, semantically representing the fact that both countries are the source of common scams against USA, and (2) a sequence of two blue $inContinent$ edges, capturing that both countries belong to a single continent as the USA. Each of the two path expressions captures a possible semantic abstraction of the attribute value, relative to the source context. The conjunction of the two path expressions captures a third abstraction representing the conjunction of the two semantic properties.

For each such abstraction, the inverse concretization function retrieves, in the ontology graph, values that are accessible by the corresponding path expression(s) from the analogous target context attribute value. For instance, when adapting the rule to a context where $country = Germany$, the concretization of the first semantic abstraction (the scams edge) includes $Orsinia$, whereas the concretization w.r.t the second semantic abstraction (same continent) includes $Orsania$ and $Qurac$. The concretization of the third semantic abstraction is the intersection of the two sets and includes only $Orsinia$, the only value connected to Germany by the two path expressions.

V. Implementation and Experiments

We have implemented the algorithms described in the previous sections in the GOLDRUSH system. GOLDRUSH is implemented in Python (backend service), PHP/JavaScript (frontend) and uses MySQL as the database engine. The system architecture is detailed in Appendix B. We next describe the datasets used in our experiments, the compared algorithms, and the experimental evaluation.

A. Datasets

We run the experiments over real-world financial transaction relations obtained from our industrial collaborators[4]. Due to the sensitivity of the financial information, we received a masked version of the datasets in which personal information such as user names, account IDs and IPs were omitted, and locations were reduced to include only the city name. We obtained five datasets belonging to five different financial institutes (FIs) around the world, for the second quarter of 2016. Each dataset consists of (i) a transactions relation, of two full months activity, and (ii) the full set of fraud detection rules for that period [5]. For each FI, we used the first month data as our training set and tested our generated rules against the second month.

Transactions and Rules: The transaction relations include payment and authentication activities performed by the FI clients. Each transaction is labeled as fraudulent/legitimate. The labeling is done as part of the standard operation of the FIs (using e.g. user notifications and periodic user approval/disproval of transactions and fraud notifications) and we use these labels as the ground truth. The relations have between 30-70 attributes, numerical attributes (such as the transaction amount, number of actions in the last hour, etc.) and categorical ones (location, client type, activity type, etc.). Each transaction also includes a risk-score attribute (a value between $0 - 1000$), generated by the machine learning module of our industrial collaborators, that indicates the probability of a transaction to be fraudulent and is usually used as one of the conditions in the rules. The rules contain between 3-20 conditions.

Figure 6 provides some additional statistics about the different datasets used for the experiments. The statistics reported in the figure are for the training set. The statistics for the test set are quite similar. The number of transactions in the relations varies from 50K to 4.2M transactions per month, where 0.1% to 2% of them are labeled fraudulent and the rest legitimate. The number of fraud detection rules for each FI varies from 44 to 110 (ignore for now the numbers in parentheses). The rule sets have precision between 0.42 and 0.86 and for all the datasets (except dataset A) the recall is often below 0.1. These

[4]Names omitted per companies request.

[5]As mentioned in Section II, the rule language that we consider, albeit simple, forms the core of common rule languages used by actual systems and covered all the obtained rules.

DS	Monthly Trx#	Fraud Ratio	# of Rules	Precision	Recall	F1 Score
A	45K	0.02	80 (27)	0.86	0.53	0.65
B	800K	0.002	44 (9)	0.65	0.0007	0.001
C	2.5M	0.01	55 (14)	0.52	0.1	0.17
D	4.2M	0.001	110 (34)	0.42	0.1	0.16
E	250K	0.001	62 (14)	0.6	0.02	0.05

Fig. 6: Datasets used for GOLDRUSH experiments

low recall numbers are common for fraud detection rules in the financial industry as fraud detection is often an extremely challenging task. Even small percentage of improvement in fraud capturing is considered a major success and yields great financial savings.

Ontology: In the experiments we used a geopolitical-financial ontology that was manually built and curated by the FIs domain experts. According to our industry collaborators such a one-time effort is cost-effective, as there are many effective rules that can benefit from the derived mappings. The ontology was built using DBPedia [11] and publicly available datasets such as CIA Factbook [8] and FATF [9] which contains lists of location-based known fraud schemes (i.e. Western Union Scams [32]), known money-laundering regions, sanctioned countries, as well as geographical properties such as each country's continent and bordering countries.

B. Algorithms

Our experiments evaluate each of the algorithms presented in the previous sections, as well as the end-to-end GOLDRUSH system. We next list the various algorithms (and competing variants) that we examine.

The GOLDRUSH *system:* In our end-to-end experiments we will use the full fledged GOLDRUSH . We use the performance of the given rule sets as baseline. Note that these rules reflect the execution of state-of-the-art ML to label the transactions with risk-scores and to suggest thresholds[6], along with the work of professional domain experts to refine the classification, and thus represent the strongest existing competitor. We denote this baseline by **ML+E**). We measure the contribution of GOLDRUSH by adding our generated rules to the FI rules set, measuring the improvement in performance, in terms of precision and recall, obtained for the extended rules set, compared to the original set. To compare to a transfer learning approach, we model the rules as decision trees and apply a state-of-the-art transfer learning algorithm from [26] (**TL**). (The construction details are given in Appendix C). Here again we add the transformed rules to the original FI rules set and examine the performance. To the best of our knowledge none of the existing decision trees TL algorithms support the incorporation of predefined mappings in the process. To nevertheless examine if/how they may improve the TL results, we also run an experiment where we first applied all possible mappings to the rules, then gave the resulting set as input to the TL algorithm (**TL**^{m}).

Best Rule Adaptation: As detailed in Section III, GOLDRUSH solves the *Best Rule Adaptation* problem by: (1) compressing the transaction relation into an *RMC Relation*,

[6]For the companies privacy, the algorithms details cannot be disclosed

Type	Method	$k=1$	$k=5$	$k=10$	$k=20$
Values	Identity	7	53	97	163
	Any-to-any	5	20	43	78
	Wildcard	4	17	28	52
	Currency	0	0	1	3
Distribution	Cum. Dist.	5	33	68	161
	Top-k Frequent	6	29	53	119
Semantic	Location Onto.	2	2	8	32

Fig. 7: Mapping Methods adoption rate

and (2) solving an ILP problem for the reduced relation. We refer below to this two steps algorithm as **GOLD**. To examine the contribution of each of these two steps, we compare *GOLD* to the following competitors:

GOLD$^-$: A restricted variant of GOLD that does not build the reduced RMC and runs the ILP on the original relation.

BF: The naïve brute-force algorithm which iterates over all possible attribute mappings, computing the score of each possible combination by issuing a corresponding SQL query over the transaction relation.

BF$^+$: Similar to *BF*, except that the queries are issued on the summarized *RMC* relation.

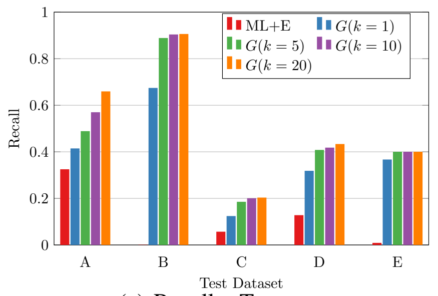
k-Rule Adaptation: Finally, we compare here our optimized k-Rule Adaptation Algorithm (Algorithm 1) to a naïve iterative solution without ILP pruning.

C. Experiments

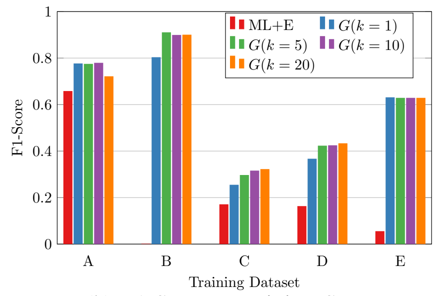
To illustrate the efficiency and effectiveness of our approach, we tested GOLDRUSH and its algorithms through three sets of experiments. We first performed an end-to-end experiment to examine GOLDRUSH as a whole, demonstrating its effectiveness for improved fraud detection and actual money saving. This also allowed us to examine which mappings are practical, thereby illustrating the usefulness of each mappings class. We next examined the runtime performance of our best rule adaptation algorithm, demonstrating its superiority compared to the competing algorithms. Finally, we compared the runtime of our optimized k-rule adaptation algorithm to the naïve iterative solution, demonstrating the effectiveness of our ILP pruning technique.

1) End-to-End Benchmark: As a recommendation system, we evaluated the quality of the recommended rule-sets proposed by GOLDRUSH in terms of statistical measures such as recall, precision, F1-score, and the amount of money that the user would save if she adopts the recommended rules. For each FI, the candidate rules pool consists of the rules set from all other FIs, and we generated k rule recommendations for $k \in \{1, 5, 10, 20\}$. The presented rule-set statistics refer to the rule set consisting of the given FI rule set union those recommended by GOLDRUSH. We run the experiments for each of the FI's training and test sets. Since the results in the two cases were similar, we present both just for the F1-score measure, and only those of the test set for the other measures.

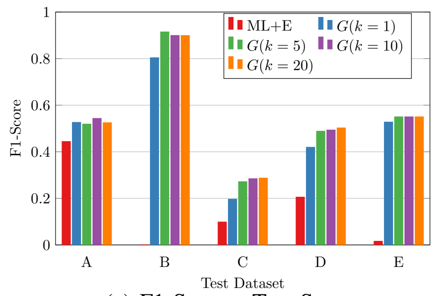
We use here the balanced ILP target score function presented in Section II-E with $\alpha = \gamma = 0.5$ and $\beta = \delta = 1$. We consider only adaptations that yield a positive cost-benefit score. Figure 6 reports (in the parentheses) for each FI the number of rules, out of all rules, for which at least one such



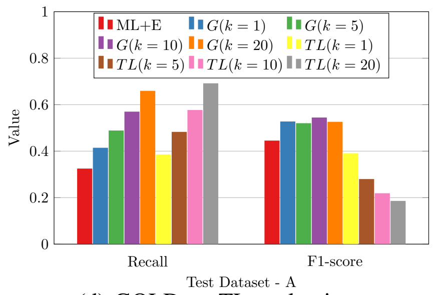
(a) Recall - Test set



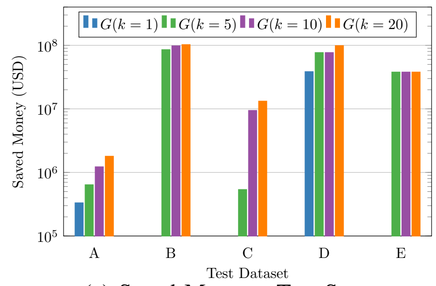
(b) F1-Score - Training Set



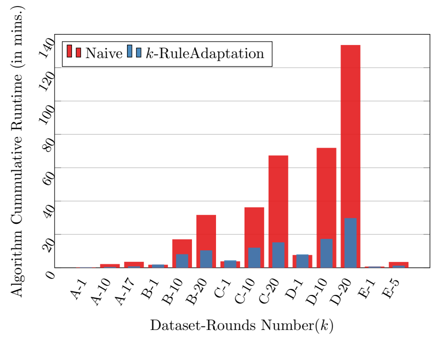
(c) F1-Score - Test Set



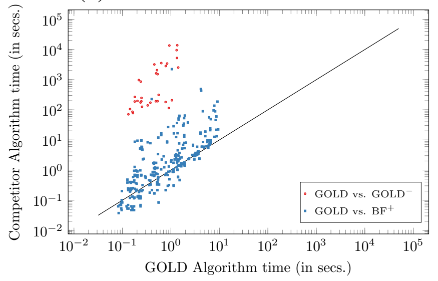
(d) GOLD vs TL evaluation



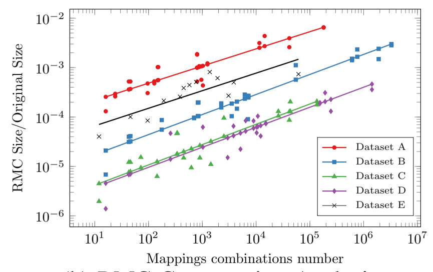
(e) Saved Money - Test Set



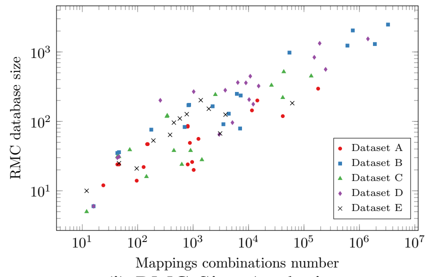
(f) k-rule adaptation algorithm performance



(g) Best Adaptation Algorithm Performance



(h) RMC Compression Analysis



(i) RMC Size Analysis

Fig. 8: Experimental results

adaptation was found in the experiments. In some cases, this implies that less than k rules would be recommended, and this is the reason why there are experiments for which the results for a given dataset remain unchanged from some k onwards. A sample of the results for GOLD vs. ML+E are depicted in Figures 8a-8c and 8e, and for GOLD vs. TL in Figure 8d.

Figure 8a depicts the recall for varying k. The first bar for each dataset shows the recall of the ML+E on the test data while the others present the recall obtained with GOLD when adding the 1, 5, 10 and 20 recommended rules. The recall naturally increases as more rules are added. For 10-rule adaptation, the recall for all datasets more than doubles, and for dataset B, where the ML+E rules set recall was extremely low, the recall raises to almost 90%. Further note that for all the datasets, an important improvement in the recall is shown already at the 1-rule adaptation, demonstrating the great benefit of even minimal rule sharing.

Regarding the precision, it increased in all the datasets (in some cases over 50%) except dataset A (where it has decreased by 18% for the large k values). Notice that to avoid such decrease, a precision threshold constraint can be added to the ILP model (not done here). We have further observed in the experiments that the precision has always increased in the 1-rule adaptation, then gradually decreased for higher k values (yet generally stays higher than that of ML+E).

Figures 8b and 8c present the F1-score for ML+E and GOLD for the training and test dataset, respectively. We use this measure to highlight the trade-off between the recall improvement and the precision decrease (which, as mentioned, occurred just for a single dataset). First, note that the results in the training and test sets are fairly similar (especially in terms of the delta between the original rules set and ours). One can also notice that, compared to the original rules set, the F1-score is improved in all cases, even for dataset A with a decreased precision.

To compare also to TL, Figure 8d depicts the recall and F1-scores for ML+E, GOLD and TL for dataset A (the results for the other datasets show similar trends). While the recall value for TL is marginally higher than GOLD for $k = 10$ and 20, the corresponding precision was extremely bad, as demonstrated by the fact that the TL F1-scores are significantly lower for all k. Morever, the rules generated by TL were not meaningful to the experts in terms of semantic understability, as illustrated in Appendix C. Interestingly, for TL^m, not only that for each input rule the algorithm now had to examine an exponential number of rules (all possible mappings) but the results did not improve relative to the original TL.

Finally, to highlight the monetary importance of our ap-

Original Rule	Translated Rule
COUNTRY $= Dinotopia \wedge$ TIME $> 20:00 \wedge$ ...	COUNTRY $= Qurac \wedge$ TIME $> 21:00 \wedge$ ...
COUNTRY $= Orsinia \wedge$ IP_CHANGED_TODAY > 3	COUNTRY $= Dinotopia \wedge$ IP_CHANGED_TODAY > 1
ACTIVITY $= Withdraw \wedge$ AMOUNT $> 5K \wedge$ ...	ACTIVITY $= Transfer \wedge$ AMOUNT $> 100K \wedge$ ...

Fig. 9: Translated Rules

proach, Figure 8e presents the amount of money that could be saved by adopting the rule sets recommended by GOLD. The amount is calculated by summing up the amounts of all the uncaptured fraudulent transactions (F_U) that were captured by some rule in the recommended set. One can see that our proposed rules might have saved between \$50K USD (for dataset A) to \$100M (!) USD (for dataset B). Looking closer at the generated rules, it is interesting to note that even when they do not save much money (as e.g. in dataset A, or in the 1st added rule in datasets B,C and E) they are still very useful, as they prevent non money-related fraudulent translations such as faulty authentication.

To conclude the experiment we examined, for each attribute mapping class, how often it was effectively used to construct the rules derived in the experiment. Interestingly all classes were useful. Figure 7 shows the results, for various mapping classes, for varying k. The most dominant mapping is the identity one, which reflects the fact that in many case values are context independent. The distribution-based is also particularly effective, as both the *cumulative distribution* mapping (used for numerical attributes) and the *top-k frequent* mapping (used for categorical attributes) were broadly chosen for the adapted rules. For the semantic-based mappings, we saw that the location-based mapping was very useful, as in more than 50% of the cases it was chosen for adapting the location-related attributes (country and city). Moreover, we noticed that the resulting recommended rules were extremely precise, with global precision (summing up all the captured transactions by all the rules) of over 95% (not shown in the table for space constraints). This supports our hypothesis that semantic analysis accompanied by a relevant ontology allows to deduce effective rule adaptations. Table 9 provides some examples of the resulting translated rules. We only show the conditions that were modified in the translation.

2) Best Rule Adaptation: In this set of experiments we compared the run-time performance of the *GOLD* algorithm against the three competitors presented in Section V-B. We show the results for the train datasets. The results for the test datasets are similar.

Single rule: We start by considering the time it takes to identify the best adaptation for a single rule. As, for a given dataset, the algorithm performance depends on the number of attribute mapping candidates, we tested the run-time for a varying number of attribute mapping combinations. In our datasets, these ranged from 10 to almost 10^7, depending on the attributes used in the given rule and their domain. Figure 8g summarizes the results of all runs, comparing the run-time of single rule adaptation performed by *GOLD* (corresponding to the x-axis) to the competitors (y-axis). Each point in the graph represents one experiment (performed on a given dataset with a given number of possible mapping combinations, and a given competitor). The point is located in the x-y coordinates representing the corresponding running time for *GOLD* and the given competitor (identified by the point color and shape).

As expected, the results of the naïve BF algorithm were significantly worse than all other algorithms and we thus omit them from the figure. The points of the GOLD$^-$ algorithm, for the dataset D, are also omitted since it took more than 3 hours even for the smallest set of attribute mapping combinations.

By the construction of the graph, points above the $y = x$ line (in black) represent cases where our GOLD algorithm outperforms its competitor. As it can be observed, most of the points are above this line. While in some cases we can see that BF$^+$ yielded better performance, a closer look at the data shows that this happens only for the smallest datasets (A and E), and with a rather small number of possible mapping combinations (less than 100), due to overhead of GOLD compared to BF. Also note that in all cases our algorithm terminated in less than 10 seconds, while for BF$^+$ some inputs required more than 100 seconds and much more the GOLD$^-$.

RMC compression: To get a better understanding for the source of efficiency of *GOLD*, compared to *GOLD*$^-$, we examined, for the different datasets, the compression ratio between the RMC relations size and that of the original relations (Figure 8h), as well as the actual RMC size (Figure 8i). In both figures we show how these number vary as a function of the number of the possible value-mapping combinations. Each experiment performed over a given dataset is represented in the figures by a dot. The dots shape and color show on which dataset the experiment was run. To highlight the compression trends in Figure 8h, we added to the figure lines that show the result of linear regression performed for each dataset. As expected, the fewer the available mappings, the greater is the compression. This is because, given a smaller set of values, the domain is split to larger intervals. Consequently more data values are represented by the same constant and can be compressed together. From the shape of the linear regression lines we can observe this trend holds for all datasets. Yet note that the compression is extremely effective even for an extremely large numbers of mappings: Even for 10^6 possible mappings, the size is reduced by over 2 orders of magnitude for dataset A, and by 3-4 orders of magnitude for the rest. Also not that the larger the dataset is, the greater is the compression ratio (for the same number of mappings). This is because more tuples are unified and can be compressed together.

We can see the actual RMC sizes in Figure 8i. Since large data sets get compressed more, the distribution of RMC sizes of all datasets is even and we get, for all datasets, RMCs of varying sizes. But in all cases they are much smaller than the original relations. As a result, even for extremely large datasets (such as dataset D with 4.2M transactions) and extremely large number of mappings (above 10^7), the data is compressed into a practical size that allows for efficient ILP solving.

k-Rules Adaptation: As mentioned, Figure 8g depicts the running times for a single rule adaptation. Given a set of candidate rules, the same process is applied to each rule in the set and the best-score rule translation is selected. In our experiments, for each FI, the candidate rules pool consists of the rule sets from all other FI's, and the running time to identify from this pool the best rule translation for FI's A, B, C, D, and E, is 0.3, 1.8, 4.4, 7.9, and 0.6 minutes, resp. (depending as expected on the size of the customer's transactions relation). Note that even for the largest dataset the time is below 8 minutes, which is very reasonable for an offline computation. More rule translations may be naturally obtained by repeating the process (see Section III-C). This is where our k-Rule threshold-based optimization comes into play, pruning unpromising rule candidates and reducing drastically the average iteration time for the FI's to 0.03, 0.4, 0.5, 1.1 and 0.15 minutes, resp.

The efficiency of our optimized *k-Rules Adaptation* algorithm (Algorithm 1) relative to the naïve iterative algorithm is demonstrated in Figure 8f, for the different datasets and for varying k. The figure presents, for each Dataset and each k, the cummulative runtimes for the *k-Rules Adaptation* and the naïve algorithms (i.e., the total runtime for calculating the first k rule recommendations). Recall that for $k = 1$ both algorithms examine all the rules in the pool and thus have similar running times. In consecutive iterations, our optimizations achieves great time reduction. We can see that already at $k = 10$ the number of ILP computations is reduced by more than a half, and the advantage typically increases as k grows. For $k = 20$, the number of ILP computations is reduced by a factor of $2.7x$ to $5x$. Note that for Datasets A and E, we present the results up to $k = 17$ and $k = 5$, resp. The reason is that best score rule in k-Rule Selection algorithm went below the threshold θ at those k values and thus no more efficient rules could be recommended from the rules pool.

VI. Related Work

The identification of fraudulent transactions is essentially a classification task which is a fundamental problem in machine learning (ML) and data management [20], [24]. As mentioned in the Introduction, fraud detection systems often employ ML and data mining techniques (e.g. [17], [22]) to score the incoming transactions. On top of this, it is common to use rules written by experts (that may include among others conditions on the ML transactions score), to refine the fraud detection and tune it to the specific company policy [25], [18]. The relationship between ML and expert-written rules has been widely discussed in the literature [6], [1], [4] for a different tasks including information extraction, conflict resolution in data integration and entity linking. Other fraud detection techniques include the use of decision tree [16], or genetic programming [3] to classify transactions into suspicious and non-suspicious ones. Our work is complementary to these lines of works and the adapted rules derived by our algorithms may be used to enhance any of these techniques.

Much of the previous research on rule-based classifiers focus on how to learn rules from the training data. Some systems, such as Chimera [29] and [18], build the rules interactively by using both ML techniques and human experts that write and refine the rules. These works however do not consider the problem of rules sharing and adaptation between different contexts. The incremental maintenance of rules in response to new incoming data has also been extensively studied in the literature (e.g. [31], [13]). In contrast our work focuses on a full context switch and employs a dedicated cost/benefit model to optimize the integration of the adapted rules to the already existing rules in the target. One technical difference is that these works considered only domains over numerical values. For such categorical values, GOLDRUSH employs dedicated mapping methods that consider e.g. value frequency as well as semantic ontology-based mappings.

Our implementation uses a geopolitical-financial ontology built with the help of our industry collaborators. Financial information about countries was also used for the task of preventing fraud in some previous work, such as [14]. Our experiments indicate that such data can indeed be effective in deriving rules with good prediction quality.

Collaboration between different parties for the improvement of fraud detection systems is sometimes enforced by local regulations (e.g. [30]). Works such as [7] present methods to encourage this collaboration and are based on centralized repositories that contain lists of fraud patterns built from fraud attacks experienced by the clients. A main drawback is that the patterns are either very general (so that they can be used by broad range of clients) and thus have low precision and/or recall, or conversely, they are specific to a certain client and require context adaptation (such as the one we provide in this paper) before they can be applicable to other clients. Combining our framework with a privacy preservation mechanism which assures that the source rule, or specific parts of it, are not discoverable from the adapted rule is an intriguing direction for future work.

Transferring knowledge from one context to another has also attracted much interest of the ML community, in areas referred to as transfer learning or domain adaptation (e.g. [21], [2]). As mentioned, closest to our work is the work on transfer learning for decision trees and random forests [26]. However, as demonstrated in our experiments, as their objective function is different the process yields rules that are less effective for the given context and often not meaningful in terms of semantics understandability.

Conceptually, the work on query reformulation (e.g., [12]) is similar to our approach. While they reformulate queries written over a source schema to a target one (under a given set of tuple/equality generating dependencies), we replace the rule conditions over a single schema using source to target condition mappings. Furthermore, our focus is on finding the best translation in terms of capturing frauds and is different from the query optimization goal which aims to minimize the size of the query. Similarly, schema matching (e.g.,[23]) can be seen as a "translation" between two schemas. However, the result of schema matching tools is a set of correspondences between the elements of the schemas, as opposed to our approach in which we map the condition values from the source and the target conditions.

VII. Conclusion

In this work we introduced GOLDRUSH, a system which facilitates knowledge sharing via effective adaptation of fraud detection rules from one context to another. Our solution employs values abstraction and concretization to map individual rule conditions from one context to another. An intuitive cost-benefit model measures the improvement in fraud detection that the resulting rules bring. While the problem of identifying the best rule adaptation is shown to be NP-hard, we employ an effective data reduction technique which, together with a dedicated ILP formulation of the reduced problem, yields a practically efficient algorithm. An extensive set of experiments on real-life datasets demonstrates the effectiveness of our approach, both in terms of the accuracy of fraud detection and the actual amount of money saved.

There are several challenging directions for future work. Specifically, we plan to extend GOLDRUSH by supporting richer rule languages, including time/event sequences, aggregates and negation. Event sequences introduce significant complications as there are potentially numerous ways to map event combinations, and pruning the search space to identify the most adequate ones may be challenging. Our mappings in this work focus on individual attributes. Considering more complex mappings that involve attribute combinations is also challenging and should take into account the dependencies among rule components. In addition, we plan to consider richer semantic mappings that may use, for example, NLP methods may be employed here to infer the semantic meaning of a condition based on textual rule description (if available). Finally, applying our rule adaptation techniques to other domains such as cyber security and medical classification is another intriguing research direction.

References

[1] B. Alexe, M. Roth, and W. Tan. Preference-aware integration of temporal data. *PVLDB*, 8(4):365–376, 2014.

[2] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Machine learning*, 79(1), 2010.

[3] P. J. Bentley, J. Kim, G.-H. Jung, and J.-U. Choi. Fuzzy darwinian detection of credit card fraud.

[4] D. Burdick, R. Fagin, P. G. Kolaitis, L. Popa, and W. C. Tan. A declarative framework for linking entities. In *ICDT*, 2015.

[5] COIN-OR CBC User Guide https://www.coin-or.org/Cbc/cbcuserguide.html.

[6] L. Chiticariu, Y. Li, and F. R. Reiss. Rule-based information extraction is dead! long live rule-based information extraction systems! In *EMNLP*, 2013.

[7] C.-C. Chiu and C.-Y. Tsai. A web services-based collaborative scheme for credit card fraud detection. In *EEE*, pages 177–181, 2004.

[8] CIA Factbook. https://cia.gov/library/publications/the-world-factbook/.

[9] Financial Action Task Force. http://www.fatf-gafi.org/.

[10] IBM CPLEX. https://ibm.com/us-en/marketplace/ibm-ilog-cplex.

[11] DBPedia. http://dbpedia.org.

[12] A. Deutsch, L. Popa, and V. Tannen. Query reformulation with constraints. *SIGMOD Record*, 35(1):65–73, 2006.

[13] F. J. Ferrer-Troyano, J. S. Aguilar-Ruiz, and J. C. R. Santos. Data streams classification by incremental rule learning with parameterized generalization. In *SAC*, pages 657–661, 2006.

[14] E. Grace, A. Rai, E. M. Redmiles, and R. Ghani. Detecting fraud, corruption, and collusion in international development contracts: The design of a proof-of-concept automated system. In *2016 IEEE International Conference on Big Data, BigData 2016, Washington DC, USA, December 5-8, 2016*, pages 1444–1453, 2016.

[15] GUROBI Optimization - ILP Solver http://www.gurobi.com/.

[16] A. I. Kokkinaki. On atypical database transactions: Identification of probable frauds using machine learning for user profiling. *Knowledge and Data Exchange, IEEE Workshop on*, 0:107, 1997.

[17] Y. Kou, C.-T. Lu, S. S, and Y.-P. Huang. Survey of Fraud Detection Techniques. *ISNSC*, 2:749–754, 2004.

[18] T. Milo, S. Novgorodov, and W.-C. Tan. Rudolf: interactive rule refinement system for fraud detection. *PVLDB*, 9(13):1465–1468, 2016.

[19] S. Mitchell, M. OSullivan, and I. Dunning. Pulp: a linear programming toolkit for python. The University of Auckland, New Zealand, 2011.

[20] T. M. Mitchell. *Machine learning*. McGraw Hill series in computer science. McGraw-Hill, 1997.

[21] S. J. Pan and Q. Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, 2010.

[22] C. Phua, V. Lee, K. Smith, and R. Gayler. A comprehensive survey of data mining-based fraud detection research. *arXiv preprint arXiv:1009.6119*, 2010.

[23] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *the VLDB Journal*, 10(4):334–350, 2001.

[24] R. Ramakrishnan and J. Gehrke. *Database management systems (3rd ed.)*. McGraw-Hill, 2003.

[25] S. Rosset, U. Murad, E. Neumann, Y. Idan, and G. Pinkas. Discovery of fraud rules for telecommunications-challenges and solutions. In *SIGKDD*, 1999.

[26] N. Segev, M. Harel, S. Mannor, K. Crammer, and R. El-Yaniv. Learn on source, refine on target: A model transfer learning framework with random forests. *TPAMI*, 2016.

[27] B. W. Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.

[28] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge unifying wordnet and wikipedia. In *WWW*, 2007.

[29] C. Sun, N. Rampalli, F. Yang, and A. Doan. Chimera: Large-scale classification using machine learning, rules, and crowdsourcing. *PVLDB*, 7(13):1529–1540, 2014.

[30] Data Sharing for the Prevention of Fraud: https://gov.uk/government/publications/data-sharing-for-the-prevention-of-fraud-code-of-practice.

[31] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1):69–101, 1996.

[32] Common types of western union scams. https://www.westernunion.com/us/en/fraudawareness/fraud-types.html.

Appendix

A. Proofs

We provide below the proofs for the theorems and propositions presented in the paper.

Proof: **[Theorem 3.1]**
The proof works by reduction from 3-SAT. Given a 3-SAT formula f with clauses $c_1, \dots, c_n$ and variables $v_1, \dots, v_m$, we build a transaction relation with attributes $A_1, \dots, A_m$ (corresponding to the variables $v_1, \dots, v_m$) each one with a 5 elements domain $D_j = \{N, T, F, \perp, \top\}$, which satisfies the partial order $N \leq F$, $N \leq T$. The transaction relation will have a legitimate transaction t_i for every clause $c_i \in f$ such that:

$$t_i[A_j] = \begin{cases} F & \text{if } v_j \in c_i \\ T & \text{if } \bar{v_j} \in c_i \\ N & \text{otherwise} \end{cases}$$

where $t_i[A_j]$ is the value of attribute A_j in transaction t_i. The rule to be adapted will be $\varphi = \bigwedge_{1 \le j \le m} A_j \le N$, with the sets of mapping candidates $V_j = \{T, F\}$. We will choose for the target function $\alpha = \beta = \delta = 0,\ \gamma = 1$, and $\theta = -0.5$ as the threshold.

Clearly, the reduction is polynomial in time. Next, we show the reduction correctness, i.e., f is satisfiable if-and-only-if there is an adaptation $\varphi' \in \Psi(\varphi)$ whose score $w(\varphi')$ exceeds the threshold θ. First of all, note that from the target function and threshold choice, it follows that such an adaptation φ' exists if-and-only-if φ' does not capture any transaction in T (since every captured transaction contributes a negative unit to the target score and we set the threshold $\theta = -0.5$).

Now, assume that f is satisfied by the assignment $< u_1, \ldots, u_m >$. Let $\varphi' = \varphi[u_1/z_1, \ldots, u_m/z_m]$ (where z_j stands for the value in A_j's condition). We will show that φ' does not capture any transaction $t_i \in T$. Let v_j be the variable corresponding to the satisfied literal in clause c_i. If the satisfied literal in c_i is v_j itself, then $u_j = T$ and $t_i[A_j] = F$, and t_i is not satisfied since $F \not\le T$. Conversely, if the satisfied literal is $\bar{v}_j$, then $u_j = F$ and $t_i[A_j] = T$, and t_i is not satisfied since $T \not\le F$.

Finally, assume that there is an adaptation in $\varphi' \in \Psi(\varphi)$ ($\varphi' = \bigwedge_{1 \le j \le m} A_j \le z_j$) that does not capture any transaction. We will now show that every clause c_i in f is satisfied by the assignment $< z_1, \ldots, z_m >$. Note that for every transaction t_i there is at least an attribute condition $A_j \le z_j$ which is not satisfied. Clearly, from the partial order definition it follows that $t_i[A_j] \neq N$. Assume that $t_i[A_j] = F$ (and thus $v_j \in c_i$). Since $A_j \not\le z_j$ it follows that $z_j = T$, satisfying literal v_j in clause c_i. Conversely, assume that $t_i[A_j] = T$ (and thus $\bar{v}_j \in c_i$). Since $A_j \not\le z_j$ it follows that $z_j = F$, satisfying literal $\bar{v}_j$ in clause c_i. ■

Proof: **[Proposition 3.5]**
First of all, note that for any sets of transactions R, S and for a rule φ the following statements are valid:

- $S \subseteq \varphi(S)$
- $\varphi(S \cup R) = \varphi(S) \cup \varphi(R)$
- If $R \subseteq S$, then $\varphi(R) \subseteq \varphi(S)$
- $\varphi(R \setminus S) = \varphi(R) \setminus \varphi(S)$
- $\varphi(R \cap S) = \varphi(R) \cap S$

Next, we bound the elements $|\varphi(F_C^i)|$, $|\varphi(F_U^i)|$, $|\varphi(L_C^i)|$, $|\varphi(L_C^i)|$ used in the formula of $w_i(\varphi)$:

$F_C^i = F_C^{i-1} \cup \varphi_{i-1}(F_U^{i-1})$. Then,

$$\varphi(F_C^i) = \varphi(F_C^{i-1}) \cup \varphi(\varphi_{i-1}(F_U^{i-1})) \subseteq \varphi(F_C^{i-1}) \cup \varphi_{i-1}(F_U^{i-1})$$

And thus,

$$|\varphi(F_C^i)| \le |\varphi(F_C^{i-1})| + |\varphi_{i-1}(F_U^{i-1})| \tag{1}$$

$F_U^i = F_U^{i-1} \setminus \varphi_{i-1}(F_U^{i-1})$. Then, $F_U^i \subseteq F_U^{i-1}$. And thus,

$$|\varphi(F_U^i)| \le |\varphi(F_U^{i-1})| \tag{2}$$

$L_C^i = L_C^{i-1} \cup \varphi_{i-1}(L_U^{i-1})$. Then, $L_C^i \supseteq L_C^{i-1} L_C$. And thus,

$$-|\varphi(L_C^i)| \le -|\varphi(L_C^{i-1})| \tag{3}$$

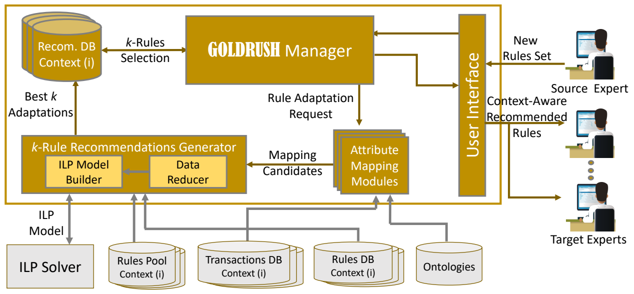


Fig. 10: GOLDRUSH architecture

$L_U^i = L_U^{i-1} \setminus \varphi_{i-1}(L_U^{i-1})$. Then,

$$\varphi(L_U^i) = \varphi(L_U^{i-1} \setminus \varphi_{i-1}(L_U^{i-1})) =$$
$$\varphi(L_U^{i-1} \cap \overline{\varphi_{i-1}(L_U^{i-1})}) = \varphi(L_U^{i-1}) \cap \overline{\varphi_{i-1}(L_U^{i-1})} =$$
$$\varphi(L_U^{i-1}) \setminus \varphi_{i-1}(L_U^{i-1})$$

And thus,

$$-|\varphi(L_U^i)| \le -|\varphi(L_U^{i-1})| + |\varphi_{i-1}(L_U^{i-1})| \tag{4}$$

Assigning (1), (2), (3) and (4) in the definition of w_i we obtain:

$$\begin{aligned} w_i(\varphi) \le \quad & \alpha \circ |\varphi(F_C^{i-1})| + \alpha \circ |\varphi_{i-1}(F_U^{i-1})| \\ & + \beta \circ |\varphi(F_U^{i-1})| \\ & - \gamma \circ |\varphi(L_C^{i-1})| \\ & - \delta \circ |\varphi(L_U^{i-1})| + \delta \circ |\varphi_{i-1}(L_U^{i-1})| \end{aligned}$$

and therefore $w_i(\varphi) \le \widehat{w}_i(\varphi)$ as required. ■

B. System Architecture

As mentioned, GOLDRUSH is implemented in Python (backend service), PHP/JavaScript (frontend) and uses MySQL as the database engine. The system architecture is depicted in Figure 10.

GOLDRUSH runs both offline and as a 24/7 service that allows for both inter and intra company collaboration. In the online mode, it continuously awaits for new rules from its clients, builds candidate rule recommendations for their collaborating parties and pushes them to the respective target clients (domain experts). The *Manager* module acts as the dispatcher for the rule adaptation task. When a new set of rules is received, it triggers the *Attribute Mapping* modules, which build the mapping candidates using the different attribute abstraction classes. These are then passed to the *k-Rule Recommendations Generator* module which runs the *k-RuleAdaptation* algorithm (Algorithm 1) to compute and store the selected rule adaptations in the *Recommended Rules DB* (to be later retrieved by the user via the UI). The algorithm invokes the *Data Reducer* module to build the relevant *RMC Relation*, then invokes the *ILP Model Builder*, which builds the ILP model and solves it using the *IBM ILOG CPLEX* ILP solver [10]. We experimented also with other ILP solvers such as GUROBI[15] and CBC[5], but they yielded inferior performance. We use the PuLP python library[19] for Linear Programming to connect to the solver.

(a) **Original rule:**

(IS_DINOTOPIA = 1) **and** (ISP_CHANGED_LAST_10_DAYS > 3)
and (TIME_ZONE = 0)

(b) **Transfer Learning Resulting rules:**

1. (TIME_ZONE > 0.5) **and** (TIME_ZONE <= 5.75)
2. (TIME_ZONE > 9.5)
3. (TIME_ZONE <= 0.5) **and** (ISP_CHANGED_LAST_10_DAYS > 3.5)

(c) **GOLDRUSH Resulting rules:**

(IS_DINOTOPIA = 1) **and** (ISP_CHANGED_LAST_10_DAYS > 4)
and (TIME_ZONE in [0, 4])

Fig. 11: Rule example

C. Transfer Learning Evaluation Results

To compare the performance of GOLDRUSH to that of the transfer learning algorithm for decisions trees in [26], we transformed each of the rules in datasets A to E into an analogous decision tree. In order to do so, we have first replaced the conditions on categorical attributes by conditions on binary attributes (e.g. *Country* = *'Dinotopia'* was translated to *'Is_Dinotopia'* = *1*), then built the decision trees top-down, each time taking the attribute that provides the highest *information gain* [26], yielding left-deep decision trees.

Applying the transfer learning algorithm in [26], we obtain for each input tree (rule) a resulting, probably complex, decision tree, which represent a set of fraud detection rules. Each rule in the set corresponding to the conjunction of the conditions along a root-to-leaf path. Finally, out of all the generated rules, we select the $top - k$ that maximize the target function presented in Section II-E.

Besides of comparing the resulting rules quality to that of GOLDRUSH (reported in Section V-C1), we have also asked five domain experts to independently examine the recommended TL rules. All experts reported that the vast majority of the rules were lacking meaningful semantics and were difficult to understand or asses. To illustrate, an example of such resulting rules is depicted in Figure 11. The original rule, written by an expert in the company which dataset A belongs to, captures knowledge about the location, time zone and number of ISP (Internet Service Provider) changes in the last 10 days. However, when adapted to the context of dataset B, the resulting TL rules were much coarser and less precise than both the original rule and the resulting GOLDRUSH adaptation (shown at the bottom of the figure). They ignore the Dinotopia country condition, and while behaving well on train data, they performed significantly worse on the test one. In contrast, the adaptation chosen by GOLDRUSH, is more specific and indeed performs well on both the train and test data.

Finally, we also tried to incorporate the mapping knowledge into the TL process. Since we are not aware of any prior work that takes such mappings into consideration, we performed the following process of enriching the TL with our mappings (we names this method $\mathbf{TL}^m$). Recall that each input rule is represented as a left deep decision tree. We processed the tree top down, generating an alternative set of trees, by replacing each rule condition by all possible attribute mapping alternatives. Consequently we obtained an exponential number of rule alternatives. We thus consider in this experiment only rules where the overall number of mapping combinations is smaller that 1000. We then run the TL on the obtained set of trees, as in the previous experiment, and examined the best generated rule from this set. Interestingly, in all experiments, the quality of the generated rule was about the same as the one obtained for standard TL (we thus omit the graph). Similarly, here again, all experts reported that the vast majority of the rules were lacking meaningful semantics and were difficult to understand or asses.