# GOLDRUSH: Rule Sharing System for Fraud Detection

Ariel Jarovsky [1] Tova Milo [1] Slava Novgorodov [1] Wang-Chiew Tan [2]
[1]Tel-Aviv University [2]Megagon Labs
[1] {arielhe, milo, slavanov}@post.tau.ac.il [2] wangchiew@megagon.ai

## ABSTRACT

Fraud detection rules, written by domain experts, are often employed by financial companies to enhance their machine learning-based mechanisms for accurate detection of fraudulent transactions. Accurate rule writing 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 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 that occur in another context. However, in practice there is a lot of commonality in what different experts are trying to achieve.

In this demo, 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 in one 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. We showcase GOLDRUSH through a reenactment of a real-life fraud detection event. Our demonstration will engage the VLDB'18 audience, allowing them to play the role of experts collaborating in the fight against financial frauds.

**PVLDB Reference Format:**
Ariel Jarovsky, Tova Milo, Slava Novgorodov, Wang-Chiew Tan. GOLDRUSH: Rule Sharing System for Fraud Detection. *PVLDB*, 11 (12): 1998-2001, 2018.
DOI: https://doi.org/10.14778/3229863.3236244.

## 1. 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 [5]. 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 on rules crafted by domain experts, in addition to the ML models.

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 context may be useful for understanding how to detect frauds in another context. The goal of GOLDRUSH is to facilitate knowledge sharing via adaptation of fraud detection rules from one context to another.

We examine the conditions of the rules and the context 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 may 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 collaborating companies, located in the USA and Germany, respectively. Figures (1) and (2) show, resp., a small portion of the transaction relations in the two companies. The relation records for each transaction the (local) time in which it was issued, its amount in local currency, the transaction type and the country from which the transaction was issued. The last column indicates whether the transactions were confirmed as fraudulent (F) or legitimate (L).*

$$\varphi^A : \textit{Type} = \text{“}Stock\ Trade\text{”} \wedge \textit{Amt} \geq 100K \wedge \textit{Time} \geq \textit{16:00} \wedge \textit{Country} \in \{Dinotopia, Jamonia\}$$

*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 score for each transaction 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 on only 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 may yield different target conditions. Observe that the condition over Time may refer to the local time or the time after the closing of the local Stock Exchange Market (i.e.,*


*Proceedings of the VLDB Endowment,* Vol. 11, No. 12
Copyright 2018 VLDB Endowment 2150-8097/18/8.
DOI: https://doi.org/10.14778/3229863.3236244.

| 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 |
| : | : | : | : | : |

Figure 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 |
| : | : | : | : | : |

Figure 2: Expert B Transactions

*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 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 $\varphi^A$ then may be the following:*

$$\varphi^B : \textit{Type} = \text{“}Stock\ Trade\text{”} \wedge \textit{Amt} \geq 95K \wedge \textit{Time} \geq 20:00 \wedge \textit{Country} \in \{Orsinia\}$$

The goal of the GOLDRUSH system 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), GOLDRUSH employs 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]. Our experiments on real-world datasets (reported in the full paper [4]) demonstrate the efficiency of our solution and the great benefits it brings in real life scenarios.

**Demonstration Overview** We will demonstrate the operation of GOLDRUSH through the reenactment of a real-life fraud detection scenario where GOLDRUSH's rule adaptation was used to stop an actual large-scale fraud attack. We will use the real (anonymized) financial transactions of three collaborating companies and demonstrate the course of the attack and its prevention: the early attack detection by one of the institutes, the (institute-specific) prevention rules written by its expects, and the rules adaptation via GOLDRUSH which allowed to block the attack also in the other (not yet suspecting) institutes. We will invite VLDB'18 attendees to play the role of domain experts in this scenario and explain the operation of GOLDRUSH through its workflow.

[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.

## 2. TECHNICAL BACKGROUND

We briefly overview the model and algorithms underlying GOLDRUSH. Full details presented in [4].

**Transaction relation** Our model consists of a relation $R(A_1, \ldots, A_m)$, called the *transaction relation*, where each tuple denotes a transaction over the attributes $A_1, \ldots, A_m$. In a financial context, each tuple denotes an operation (i.e. transfer, payment) made through some bank or credit card. The transaction relation is appended with more transactions over time. Attributes can have a *numerical* domain (i.e. $Amount$ in example 1.1) or a *categorical* one (i.e. $Country$). Each domain may have a semantic partial order associated with it. A transaction may be classified *fraudulent* which means that the transaction was carried out illegally. Otherwise, it will be classified as *legitimate*. For simplicity we will assume that all the collaborating parties use the same relation schema. Otherwise a schema matching component can be added.

**Rules** Fraud detection experts specify rules that can be applied to a transaction relation to discover fraudulent transactions. For simplicity and efficiency of execution, the rules are typically written over a single relation, which is a universal transaction relation that includes all the necessary attributes (possibly aggregated or derived from many other database relations) for fraud detection [6]. Hence, it is not necessary to consider explicit joins over different relations in the rule language. A rule $\varphi$ is defined as a conjunction of conditions over the attributes of the transaction relation, i.e., is is of the form $\varphi = \bigwedge_{1 \leq i \leq m} \alpha_i$ where $\alpha_i$ is a condition of the form '$A_i\ op_i\ z_i$' , $op_i \in \{=, \neq, <, >, \leq, \geq, \in, \notin\}$. 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[6].

More formally, if $\varphi$ is a rule that is specified over a transaction relation $R$, then $\varphi(R)$ denotes the set of all tuples in relation $R$ that satisfies $\varphi$. We say that $\varphi(R)$ are the transactions in $R$ that are *captured* (and suspected as fraud) by $\varphi$. If $\Phi$ denotes a set of rules over $R$, then $\Phi(R) = \bigcup_{\varphi \in \Phi} \varphi(R)$. In other words, $\Phi(R)$ denotes the union of results of evaluating every rule in $\Phi$ over $R$.

**Attribute-mappings** 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 $z_i$ used in the rule conjuncts by those that best match the target context. As we will see, such value-based mappings are extremely effective.

More formally, given a rule $\varphi = \bigwedge_{1 \leq i \leq m} A_i\ op_i\ z_i$, defined by an expert for some source context $c_s$ (consisting of the company transactions relation, existing rules and other data regarding the company), we want to map it into a rule $\varphi'$ for a target context $c_t$. An *abstraction-based attribute mapping* is a pair $(g, h)$ that consists on an *abstraction* function ($g$) and a *concretization* function ($h$). The function $g$ provides the way to generalize an attribute value $z$ in $\varphi$ from its concrete value in $c_s$ to a general concept that should be semantically meaningful independently of the context.

$$S_{Amount} = \{100K(\textbf{\textit{ID}}), 95K(\textbf{\textit{CC}}), 200K(\textbf{\textit{VP}})\}$$
$$S_{Time} = \{16{:}00(\textbf{\textit{ID}}), 20{:}00(\textbf{\textit{CT}}), 19{:}00(\textbf{\textit{VP}}), 22{:}00(\textbf{\textit{LT}})\}$$
$$S_{Country} = \{\{\text{Dinotopia, Jamonia}\}(\textbf{\textit{ID}}), \{\text{Orsinia}\}(\textbf{\textit{MA}})\}$$
$$S_{Type} = \{\text{"Stock Trade"}(\textbf{\textit{ID}})\}$$

Figure 3: Translation values for $\varphi^A$

The function $h$ takes a generalized concept and maps it to a concrete attribute value in $c_t$.

For an attribute $A_i$ in $\varphi$ with value $z_i$, and a set of possible abstraction-based attribute mappings $(g_{i_1}, h_{i_1}), \ldots, (g_{i_k}, h_{i_k})$ for $A_i$, the set $S_i = \bigcup_{1 \le j \le k} h_{ij}(g_{ij}(z_i))$ represents the possible value translations for $z_i$. Then a possible translation of $\varphi$ to context $c_t$ is a rule a of the form $\varphi' = \bigwedge_{1 \le i \le m} A_i \; op_i z'_i$, where $z'_i \in S_i$.

EXAMPLE 2.1. *Continuing with example 1.1, we can use different abstraction-based attribute mappings for each one of the attributes. Figure 3 shows some of the possible mapped values for the attributes of $\varphi^A$ (after applying both the relevant abstraction and concretization functions). For the amount attribute, one possible mapping is currency conversion - the abstraction function here maps the local amount to some general agreed upon currency; a conversion to Euro is then used as the concretization function (marked in the figure as **CC**). Another distribution-based mapping maps the value to its corresponding percentile (i.e. upper 5%) in the amounts appearing the transactions in context A. The concretization function then maps the abstract percentile to a concrete value threshold in the target context B (marked in the figure as **VP**). For the time attribute, we may use in a similar manner local time conversion (**LT**) or distribution based conversion (**VP**). Alternatively, we can use a semantics abstraction that maps it to the Stock Market closing time concept, with mapping to Frankfurt Stock Market closing time for concretization (**CT**).*

**Cost & Benefit model** To compare between the different translation candidates and determine which is the most suitable one, 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). We omit the formal definition of the cost-benefit formula here, but note that our goal is identify the translation that maximizes its value.

**The main algorithms behind GOLDRUSH** There are two main challenges to overcome to yield good rule translations: (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 translations. We will first discuss (2), assuming that the set of possible mappings for every attribute value is given, then explain what mappings are used.

*Identifying optimal translations.* As mentioned in the introduction, finding the rule translation that maximizes the cost-benefit score can be shown to NP-hard. However, the problem may be modeled as an Integer Linear Programming (ILP) problem. While this still NP-hard, ILP solvers are known to be efficient in practice for a not too large number of variables. As this not the case in our setting (the variables in the ILP formulation correspond to the number of transactions, which may be millions in common scenarios), we first employ a dedicated preprocessing data-reduction step. Our technique clusters together transactions that "behave uniformly" w.r.t the conditions in the rule, representing them as a single tuple, thereby significantly reducing the ILP problem space and yielding efficient performance.

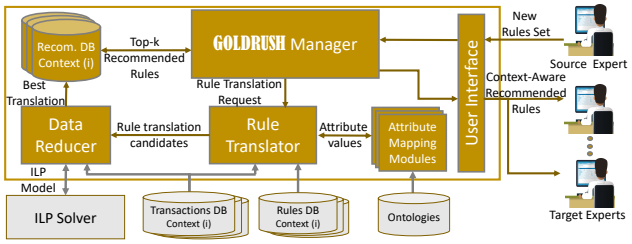


Figure 4: GOLDRUSH architecture

Intuitively, our reduced ILP problem consists of a 0-1 variable for every value in each $S_i$ (with the value 1 symbolizing that the corresponding translation was chosen) and a 0-1 variable corresponding to every transactions cluster tuple (with the value 1 symbolizing that the cluster is captured by the chosen translation). The objective function allows to weight the benefit of the correctly identified fraudulent transactions vs. the cost of the misclassified legitimate transactions. We also have a constraint for every attribute $A_i$ in the rule, ensuring that a single value is selected from $S_i$, and a pair of constraints that ensure that the cluster variables are set to 1 if-and-only-if all the clauses of the rule are satisfied by the chosen combination of selected values. Our solution also provides the user with the option to add more specific requirements such as a precision threshold, misclassification threshold, etc.

*Building Attribute mappings* We employ in GOLDRUSH three classes of mappings that may be used. The first class consists of a standard set of value-based mappings. These include for instance currency, temperature, time zone, length and weight metrics mappings. The identity mapping also belongs to this set, and so is a special "True" mapping (that allows to remove conditions from the rule). The second class includes distribution-based mappings such as percentile and frequency (top/bottom-k). The third class is data driven and employs an ontological knowledge base (constructed by importing data from DBPedia [2]). To determine possible abstractions, the semantic properties that the attribute values satisfy in relation to the given context, are mined. For instance, in our running example we can see that all countries mentioned in the Country attribute are related to the US by a *common-scam* relation. The concretization mapping then maps the identified pattern to its instantiation in the target context (the set of countries that are related to Germany by the same *common-scam* relation). This module is extensible and additional methods for identifying relevant abstractions/concretizations may be added. As an example we implemented an interactive module that consults the experts to get additional mappings.

## 3. SYSTEM AND DEMONSTRATION

GOLDRUSH is implemented in Python (backend service), PHP/JavaScript (frontend) and uses MySQL as the database engine. The system architecture is depicted in Figure 4. The *Manager* module acts as the dispatcher for the rule translation task. When a new set of rules is received, it triggers the *Rule Translator* module, which builds the rule translation candidates set and pass it to *Attribute Mapping modules*. Once the relevant attribute mappings are computed, they are sent to the *Data Reducer* which clusters the relevant transactions relation, builds the corresponding ILP model and solves it using the 3rd-party libraries ([1], [7]). Finally, the best rule translations are added to the *Recommended Rules DB* from where they are continuously pushed to the experts through the UI (as pop ups or upon request).

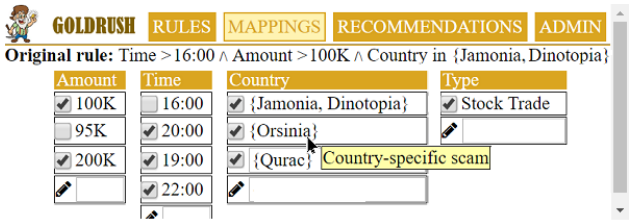


Figure 5: GOLDRUSH UI: Mappings (concretization)

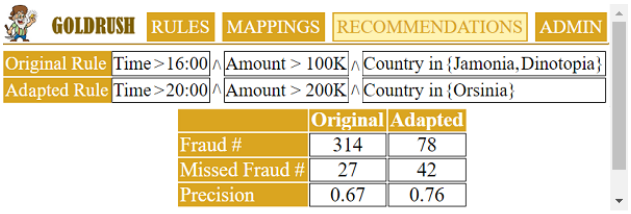


Figure 6: GOLDRUSH UI: Recommended Translations

**Demonstration scenario** As mentioned, we will demonstrate the functionalities of GOLDRUSH through a reenactment of a real-life fraud detection scenario where GOLDRUSH's rule adaptation is used to stop a large-scale fraud attack. In the demonstrated scenario, an online attack is launched against gas-stations' convenience stores of different chains in several countries, issuing small-amount fraudulent money-transfer transactions around the stores' closing time. The attack is hard to detect as the transactions are of varying amounts and are are distributed, with only few transactions per store, across multiple stores in the chain. Due to this difficulty, only one company identified the incident, battling it by (chain-specific) prevention rules written by its domain expects. It is only the rules adaptation via GOLDRUSH which allowed to block the attack also in the other (not yet suspecting) companies, saving large amounts of money.

We will run the demonstration over a snapshot of three real-world transaction relations obtained from our industrial collaborators, belonging to to companies $A$, $B$ and $C$ (identity is not given for privacy considerations) from the above scenario. The transaction datasets contain attributes similar to the ones presented in Figure 1 and more. We will use a masked version of the datasets in which sensitive information such as user/company names, account IDs and IPs are masked. Some of the transactions are identified as fraudulent and others as legitimate. We will start the demonstration by presenting GOLDRUSH, its interface and the system's main goal, then briefly describe the scenario that we are about to reenact, and the context characteristics (country, currency, work hours, etc.) of each of the three chains. We will use in the demo three laptops, one per company and show the users the incoming flow of the company transactions, with the reported fraudulent transactions highlighted. We will also explain what made the experts of company $A$ suspect the individual fraudulent transactions to belong to a coordinated attack.

We will then ask the users to play the role of the experts and add, through the system UI, rules to prevent the attack in company $A$. For users lacking relevant background we will provide a cheat sheet with the actual rules written by the company experts, which they can copy/modify. A screen with a visualization of the different attribute value abstractions will then be shown, allowing the user to mark the more relevant ones, if desired (as shown in Figure 5). The system will then analyze whether the rules may be shared by the other companies and adapted to their context. Whenever successful, a rule suggestion will pop-up on the corresponding laptop, also showing the different concretizations built for the attribute values. Once the suggested rules are accepted, we will see how the incoming fraudulent transactions are identified and the attack is blocked. Importantly, the audience will be able to observe how the adaptation of the same $A$ rule differs in companies $B$ and $C$, following their specific context.

To illustrate what happens behind the scene, we will also allow the audience to examine the different steps of our solution: the compressed tuple clusters (compared to the original tuples set they represent), the reduced-size ILP model, and finally the recommended rule displayed to the users, along with relevant information such as the particular semantic abstractions employed, statistics on the number of fraudulent transactions captured/missed by the rule at the target context, its precision and recall, etc. (Figure 6). We also provide for comparison similar statistics regarding the performance of the original rule in its source context.

**Related work** Fraud detection typically employs ML and data mining methods (e.g., [5]). As the main focus of our work is translating rules from one context to another (by using different translation methods), it is closely related to transfer learning (e.g. [8]). However as demonstrated in the full paper ([4]), their target function is different and the resulting rules that are less effective for the given context and often not meaningful in terms of semantics and readability.

Conceptually, the work on query reformulation (e.g., [3]) is close to ours. But 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 objective function to best capture frauds is different from the query optimization goal which aims to minimize the size of the query. Similarly, schema matching (e.g.,[9]) can be seen as a "translation" between two schemas. However, the result of schema matching tools is a set of correspondences between schema elements, while we map the condition values from the source and the target conditions. Finally, existing work on incremental maintenance of rules in response to new incoming data (including own work in [6]) performs only an optimization of the existing rules *within a given single context* and does not consider knowledge sharing in multiple contexts. The two lines of works are thus complimentary.

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

# 4. REFERENCES

[1] IBM CPLEX. https://ibm.com/us-en/marketplace/ibm-ilog-cplex.
[2] DBPedia. http://dbpedia.org.
[3] A. Deutsch, L. Popa, and V. Tannen. Query reformulation with constraints. *SIGMOD Record*, 35(1):65–73, 2006.
[4] A. Jarovsky, T. Milo, S. Novgorodov, and W.-C. Tan. Rule sharing for fraud detection via adaptation. In *ICDE*, 2018.
[5] Y. Kou, C.-T. Lu, S. S, and Y.-P. Huang. Survey of Fraud Detection Techniques. *ICNSC*, 2:749–754, 2004.
[6] T. Milo, S. Novgorodov, and W.-C. Tan. Interactive Rule Refinement for Fraud Detection. In *EDBT*, 2018.
[7] S. Mitchell, M. OSullivan, and I. Dunning. Pulp: a linear programming toolkit for python. The University of Auckland, New Zealand, 2011.
[8] S. J. Pan and Q. Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, 2010.
[9] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *the VLDB Journal*, 10(4):334–350, 2001.