Rock: Cleaning Data with both ML and Logic Rules

Zian Bao*
Binbin Bie
baozian@sics.ac.cn
biebinbin@sics.ac.cn
Shenzhen Institute of Computing Sciences, China

Wenfei Fan
Shenzhen Institute of Computing Sciences, China
University of Edinburgh
United Kingdom
Beihang University, China
wenfei@inf.ed.ac.uk

Daji Li
Mengyun Li
lidaji@sics.ac.cn
limengyun@sics.ac.cn
Shenzhen Institute of Computing Sciences, China

Kaiwen Lin
Wei Lin
linkaiwen@sics.ac.cn
linwei@sics.ac.cn
Shenzhen Institute of Computing Sciences, China

Peijie Liu
Peng Liu
liupeijie@sics.ac.cn
liupeng@sics.ac.cn
Shenzhen Institute of Computing Sciences, China

Zhicong Lv
Mingliang Ouyang
lvzhicong@sics.ac.cn
ouyangmingliang@sics.ac.cn
Shenzhen Institute of Computing Sciences, China

Chenyang Sun
Shuai Tang
sunchenyang@sics.ac.cn
tangshuai@sics.ac.cn
Shenzhen Institute of Computing Sciences, China

Yaoshu Wang
Qiyuan Wei
yaoshuw@sics.ac.cn
werty@sics.ac.cn
Shenzhen Institute of Computing Sciences, China

Xiangqian Wu
Min Xie
wuxiangqian@sics.ac.cn
xiemin@sics.ac.cn
Shenzhen Institute of Computing Sciences, China

Jing Zhang
Runxiao Zhao
zhangjing@sics.ac.cn
zhaorunxiao@sics.ac.cn
Shenzhen Institute of Computing Sciences, China

Jie Zhu
Yilin Zhu
zhujie@sics.ac.cn
zhuyilin@sics.ac.cn
Shenzhen Institute of Computing Sciences, China

ABSTRACT

We demonstrate Rock, a system for cleaning relational data. Rock highlights the following unique features: (1) it extends logic rules by embedding machine learning models as predicates, to benefit from both ML and logic deduction; (2) it supports entity resolution, conflict resolution, timeliness deduction and missing data imputation in a unified process; and (3) it provides parallelly scalable algorithms for rule discovery, error detection and error correction, in batch and incremental modes. We will demonstrate Rock for its (a) easy-to-use interface, (b) scalability when cleaning large datasets, (c) accuracy for detecting and correcting errors across multiple tables, and (d) applications at banks and HR departments.

PVLDB Reference Format:
Zian Bao, Binbin Bie, Wenfei Fan, Daji Li, Mengyun Li, Kaiwen Lin, Wei Lin, Peijie Liu, Peng Liu, Zhicong Lv, Mingliang Ouyang, Chenyang Sun, Shuai Tang, Yaoshu Wang, Qiyuan Wei, Xiangqian Wu, Min Xie, Jing Zhang, Runxiao Zhao, Jie Zhu, and Yilin Zhu. Rock: Cleaning Data with both ML and Logic Rules. PVLDB, 17(12): 4373 - 4376, 2024.
doi:10.14778/3685800.3685878

1 INTRODUCTION

Dirty data has been a longstanding challenge. Real-life data is often dirty, evidenced by duplicates, conflicts, missing data and obsolete values commonly found in our datasets. Data-driven decisions based on dirty data can be worse than making decisions with no data.

In light of these, data cleaning systems have been studied for decades. Nonetheless, when we work with practitioners in industry, we often hear the following questions and requests.

• *Machine learning (ML) or logic deduction?* Existing systems approach data quality typically via either logic rules or ML models. However, none of the two is super to the other. On the one hand, it is hard to find a small number of logic rules that cover all errors and data in practice. On the other hand, ML solutions are probabilistic and hard to interpret; practitioners may not want to deploy ML models when cleaning critical data such as medical records.

Is it possible to unify ML and logic rules in the same process, to benefit from both? How effective can such a uniform framework be, compared to logic deduction and ML predictions alone?

• *Functionality.* Data cleaning systems have mostly focused on two primitive issues: (a) *entity resolution* (ER), to determine whether two tuples refer to the same real-world entity, in order to catch duplicates; and (b) *conflict resolution* (CR), to detect and resolve semantic inconsistencies among attribute values of the entities.

However, there are two other critical issues of data quality: (c) *missing data imputation* (MI), to enrich tuples by filling in the missing values (null); and (d) *timeliness deduction* (TD), to deduce temporal orders on attribute values, and infer the latest values of each entity. The need for addressing these is evident in practice. For example, 42.5% of epidemiological records are incomplete [13], and "58% of organizations make decisions based on outdated data" [4].

Moreover, these critical issues interact with each other. On the one hand, deducing missing values and temporal orders help us identify entities and fix inconsistencies. On the other hand, ER and

*Author names are listed in alphabetical order.

Proceedings of the VLDB Endowment, Vol. 17, No. 12 ISSN 2150-8097.
doi:10.14778/3685800.3685878

CR facilitate us to enrich tuples by instantiating missing values and deduce timeliness by providing more correlated values.

Is it possible to support all of ER, CR, MI and TD in a system? Can we leverage their interactions to improve the overall data quality?

• *Performance.* To clean dirty data, a cleaning system should support the following: (a) *rule discovery*, to learn logic rules and/or train ML models; (b) *error detection*, to catch errors (duplicates, inconsistencies, missing and stale values) with the rules/models learned; and (c) *error correction*, to fix the errors detected (merge duplicates, resolve conflicts, fill in null values and deduce the latest values).

Can we have a system that scales with large datasets? Can it ensure that its error correction has the "certainty", *i.e.,* it fixes an existing error and introduces no new errors?

Rock (Section 2). To answer these questions, we have developed Rock [1, 3], a system for cleaning relational data. Rock has been deployed at banks, delivery services, mobile operators and e-commerce, and proven effective there. It is unique in the following.

(1) ML in logic rules. Rock proposes REE^{++}s, an extension of Entity Enhancing Rules (REEs) [9, 11]. REE^{++}s (a) embed ML models as predicates, to benefit from both ML predictions and logic deduction, and (b) express rules for ER, CR, TD [12] and MI [8], and subsume conditional functional dependencies (CFDs) [7], denial constraints (DCs) [2] and matching dependencies (MDs) [5] as special cases.

(2) A unified process. Rock supports ER, CR, TD and MI in the same process, and leverages their interactions to improve the overall quality of the data. It provides algorithms for rule discovery [9, 10], error detection and error correction [8, 11], in batch and incremental modes [11], to deal with static and dynamic datasets, respectively.

(3) Scalability and accuracy. The algorithms underlying Rock are provably parallel scalable [14], *i.e.,* they guarantee to reduce runtime when more machines are used, and hence in principle, can scale with large datasets. Rock also develops other optimization strategies, *e.g.,* top-k discovery to reduce excessive and redundant rules [10]. Moreover, its fixes to errors are logical consequences of the rules employed, ground truth accumulated and ML predictions; if the rules, ground truth and predictions are accurate, so are the fixes.

Demonstration (Section 3). Participants can interact with Rock to experience its (a) friendly user interface, (b) accuracy improvement from its unification of ML and logic deduction and the interactions among ER, CR, TD and MI; and (c) capability to scale with large datasets ensured by the parallel scalability and optimization strategies of its underlying algorithms. We will also showcase how Rock works in regularity reporting and human resource management.

2 AN OVERVIEW OF ROCK

This section presents the foundation and architecture of Rock [3].

2.1 Foundation of Rock

Rock is based on the class of REE^{++}s, which are defined over a database schema $\mathcal{R} = (R_1, \ldots, R_m)$, where R_j is a relation schema $R(A_1, \ldots, A_k)$, and each A_i is an attribute of relation R.

An REE^{++}s (Entity Enhancing Rule) over schema $\mathcal{R}$ is defined as

$$\varphi : X \rightarrow p_0,$$

where X is a conjunction of *predicates* over $\mathcal{R}$, and p_0 is a predicate

Table 1: Example Drug **relation** D_1

tid	did	fid	Name	Spec	Descr	Admin
t_1	d_1	f_1	Inosine Oral	20ml*10;0.2g*10	for hepatitis ...	OA
t_2	d_2	f_2	Inosine	200ml;20g	Hepatitis disease ...	OA
t_3	d_3	f_3	Finasteride	5mg	treat BPH ...	OA

Table 2: Example Factory **relation** D_2

tid	fid	CName	Addr	Legal	Annual (M)	CSize	Type
t_4	f_1	B. medicine	12 Beijing Str.	null	2	30	micro
t_5	f_2	B. medicine	15 Nanjing	null	2.5	36	**large**
t_6	f_2	B.	14 Beijing Str.	null	2.5	37	small
t_7	f_2	medicine	14 Beijing Str.	Q. Zhang	2.5	37	null
t_8	f_3	medicine	14 Beijing Str.	Q. Zhang	2.5	37	small

over $\mathcal{R}$ such that its tuple variables are bounded in X (see below).

Predicates over schema $\mathcal{R}$ include the following:

$$p ::= R(t) \mid t.A \oplus c \mid t.A \oplus s.B \mid t \otimes_A s \mid p_G \mid \mathcal{M}(t[\bar{A}], s[\bar{B}])$$

Here (a) $R(t)$ is a *relation atom* over $\mathcal{R}$, where $R \in \mathcal{R}$, and t is a *tuple variable bounded by* $R(t)$; (b) $t.A \oplus s.B$ compares *compatible* attributes $t.A$ and $s.B$, where $\oplus$ is one of $=, \neq, <, \leq, >, \geq$, tuple t (resp. s) is bounded by $R(t)$ (resp. $R'(s)$), and $A \in R$ and $B \in R'$ have the same type; similarly for $t.A \oplus c$; (c) $\otimes$ is either $\prec$ or $\preceq$, and $t \prec_A s$ says that the value of attribute $s[A]$ is more up-to-date than $t[A]$; similarly for $t \preceq_A s$; (d) p_G denotes predicates for extracting values from a knowledge graph and filling in missing values of a tuple (see [8] for details); and $\mathcal{M}(t[\bar{A}], s[\bar{B}])$ is *an ML predicate*, where $t[\bar{A}]$ and $s[\bar{B}]$ are vectors of pairwise compatible attributes.

ML predicates. REE^{++}s may embed ML models $\mathcal{M}$ as predicates. Here $\mathcal{M}$ can be any black-box function that returns a Boolean value, for classification, similarity checking, link prediction, and error detection, *e.g.,* $\mathcal{M}_{\mathsf{reg}} \geq \delta$ for the strength of a regression model $\mathcal{M}_{\mathsf{reg}}$ and a predefined threshold δ. In particular, Rock trains several models for temporal ranking [12] and assessing correlation between a (partial) tuple and another attribute [8]; *e.g.,* $\mathcal{M}_{\mathsf{rank}}(t_1, t_2, \prec_A)$ returns true if it predicts $t_1 \prec_A t_2$, and false otherwise.

As an example, consider a drug database schema $\mathcal{R}$ with relations in Tables 1 and 2. Below are two REE^{++}s over schema $\mathcal{R}$.

φ_1 : $\mathsf{Drug}(t) \wedge \mathsf{Drug}(s) \wedge \mathcal{M}_{\mathsf{norm}}(t.\mathsf{Spec}, s.\mathsf{Spec}) \wedge \mathcal{M}_{\mathsf{sim}}(t.\mathsf{Name}, s.\mathsf{Name}) \rightarrow t.\mathsf{Admin} = s.\mathsf{Admin}$, where $\mathcal{M}_{\mathsf{sim}}$ is a semantic matching model to identify drug names, and $\mathcal{M}_{\mathsf{norm}}$ normalizes the specifications of drugs and checks whether the capacities and weights of two drugs are the same. The rule says two drugs have the same drug administration if they have similar names and same specifications.

φ_2 : $\mathsf{Factory}(t) \wedge \mathsf{Factory}(s) \wedge X \rightarrow \mathcal{M}_{\mathsf{add}}(t[\mathsf{Addr}], s[\mathsf{Addr}])$, where $\mathcal{M}_{\mathsf{addr}}$ checks whether two addresses are the same, $X = \bigwedge_{A \in \mathcal{T}} t.A = s.A$ and $\mathcal{T}$ is a set of attributes about factory addresses (not shown), *e.g.,* zipcode and neighborhood information. Rock discovers conditions in X to explain why $\mathcal{M}_{\mathsf{addr}}$ predicts true.

A full range of ER, CR, TD and MI. REE^{++}s are able to express rules for the four central issues of data quality. As an example, φ_1 above is an REE^{++}s for CR. Below are more REE^{++}s for CR and ER:

φ_3 : $\mathsf{Factory}(t) \wedge \mathsf{Factory}(s) \wedge t.\mathsf{fid} = s.\mathsf{fid} \wedge \mathcal{M}_{\mathsf{range}}(t[\mathsf{Annual}, \mathsf{CSize}], s[\mathsf{Annual}, \mathsf{CSize}]) \rightarrow t.\mathsf{Type}{=}s.\mathsf{Type}$, where $\mathcal{M}_{\mathsf{range}}$ checks whether the annual revenues and the employee numbers of two factories are in the same range based on the national standard. The rule states that if records t and s are for the same factory, and have similar revenues and employee numbers, then the two have the same type.

φ_4 : $\mathsf{Drug}(t) \wedge \mathsf{Drug}(t') \wedge \mathsf{Factory}(s) \wedge \mathsf{Factory}(s') \wedge \mathcal{M}_{\mathsf{sim}}(t.\mathsf{Descr}, t'.\mathsf{Descr}) \wedge t.\mathsf{Admin} = t'.\mathsf{Admin} \wedge t.\mathsf{fid} = s.\mathsf{fid} \wedge t'.\mathsf{fid} =$

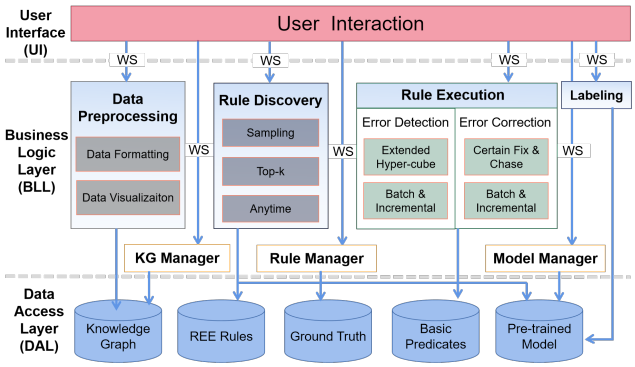

Figure 1: Rock architecture

$s'.\text{fid} \wedge s.\text{CName} = s'.\text{CName} \wedge s.\text{Type} = s'.\text{Type} \rightarrow t.\text{did} = t'.\text{did}$, where $\mathcal{M}_{\text{sim}}$ checks the semantic similarity between the descriptions of two drugs. It states that two drug records refer to the same one if they have the similar descriptions, the same drug administrations and they are produced by the same factory. Unlike CFDs and DCs, φ_4 involves *four* tuples in *two* tables, beyond bi-variable rules.

REE^{++}s with temporal predicates can express interesting properties, *e.g.,* monotonicity, comonotonicity and correlation, for TD.

$\varphi_5 : \text{Factory}(t) \wedge \text{Factory}(s) \wedge t.\text{fid} = s.\text{fid} \wedge t.\text{Type} = \text{“micro”} \wedge s.\text{Type} = \text{“small”} \rightarrow t \preceq_{\text{Type}} s$. It says the types of the factory typically changes *monotonically*, *i.e.,* from “micro” to “small”.

$\varphi_6 : \text{Factory}(t) \wedge \text{Factory}(s) \wedge t.\text{fid} = s.\text{fid} \wedge t \preceq_{\text{Type}} s \rightarrow t \preceq_{\text{Addr}} s$, *i.e.,* $\preceq_{\text{Type}}$ and $\preceq_{\text{Addr}}$ are often comonotonic: when the type of a factory changes, then the address of this factory may also change.

The following REE^{++}s impute missing values for MI.

$\varphi_7 : \text{Factory}(t) \wedge \text{vertex}(x, \text{KG}) \wedge \text{HER}(t, x) \wedge \text{match}(t[\text{Legal}], x.(\text{legal})) \rightarrow t[\text{Legal}] = \text{val}(x.(\text{legal}))$. It says that if a factory t matches a vertex x in a KG of enterprise and if x reaches vertex v via path $\rho = (\text{legal})$, let $t[\text{Legal}]$ take $L(v)$ as its value. Here $\text{HER}(t, x)$ is a predicate that returns true if a tuple t in a relation and a vertex x in a graph refer to the same real-world entity, by heterogeneous entity resolution across relations and graphs [6].

$\varphi_8 : \text{Factory}(t) \wedge \text{Factory}(t') \wedge \text{Factory}(t'') \wedge t.\text{fid} = t'.\text{fid} \wedge t.\text{fid} = t''.\text{fid} \wedge t[\text{Addr}] = \text{null} \wedge t' \preceq_{\text{Addr}} t'' \rightarrow t.\text{Addr} = t''.\text{Addr}$. It fills in the null Addr of t by a more recent address.

The interaction. Better still, Rock makes use of the interaction among ER, CR, TD and MI, to improve the overall quality. It can also automatically generate a sequence of decision-making processes for any subset of functionalities above [3]. Consider Tables 1-2.

(1) ER helps CR. Applying φ_4 to (t_1, t_2) of D_1, we identify d_1 and d_2, *i.e.,* ER. By applying another rule $\varphi_9 : \text{Drug}(t) \wedge \text{Drug}(s) \wedge t.\text{did} = s.\text{did} \rightarrow t.\text{fid} = s.\text{fid}$, we deduce that f_1 and f_2 are the same factory. Then applying φ_3 of CR to (t_4, t_5) of D_2, we fix the erroneous type of t_5 (shown in bold) by letting $t_5[\text{Type}]$ =“micro”.

(2) CR helps TD. Once the errors in Type are fixed by CR, we can rank the timeliness for tuples $t_4 - t_6$ in D_2. For example, by applying φ_5 and φ_6, we can deduce that “small” and “14 Beijing Str.” are the latest Type and Addr of the factory, respectively.

(3) TD helps MI. After identifying the latest type “small” of t_6, we apply φ_8 to (t_6, t_7) in D_2, and impute the missing type of t_7 as “small”.

(4) MI helps ER. After knowing the type of t_7, we apply REE^{++}s $\varphi_{10} : \text{Factory}(t) \wedge \text{Factory}(s) \wedge t.\text{Legal} = s.\text{Legal} \wedge t.\text{Addr} = s.\text{Addr} \wedge t.\text{Type} = s.\text{Type} \rightarrow t.\text{fid} = s.\text{fid}$ to (t_7, t_8), to identify f_2 and f_3 (ER).

Remark. (1) Popular data quality rules such as DCs, MDs and CFDs are special cases of REE^{++}s [11]. For example, φ_9 can be written as a DC, MD or CFD, and is expressed an REE^{++}. (2) As opposed to previous rules, REE^{++}s embed $\mathcal{M}$ as predicates in precondition X (see φ_1), and moreover, can uniformly express rules for ER, CR, TD and MI. (3) As shown by φ_2, for certain ML models $\mathcal{M}$, Rock can discover logic conditions X to provide high-level rational behind predictions of $\mathcal{M}$, via REE^{++} of the form $X \rightarrow \mathcal{M}(t[\bar{A}], s[\bar{B}])$.

2.2 The Architecture of Rock

As shown in Figure 1, Rock is developed based on a three-tier architecture. User interface (UI) is the topmost level of the architecture. It displays standard graphical interfaces, receives user requests, communicates with other layers via Web socket (WS) and returns the results to users. The business logic layer (BLL) conducts processing. It also moves and processes data between the two surrounding layers. The data access layer (DAL) provides APIs for BLL to access and manage the stored data. The retrieved data is passed back to BLL for processing, and is eventually returned back to the users.

Given a dataset $\mathcal{D}$ of schema $\mathcal{R}$, Rock automatically discovers a set Σ of REE^{++}s over $\mathcal{R}$ *offline* [9, 10]. It then detects and fixes errors in $\mathcal{D}$ online, using the REE^{++}s in Σ. Moreover, the users may use Rock to monitor changes to $\mathcal{D}$, and incrementally detect and fix errors in response to updates. Rock also accumulates a library of pre-trained ML models with various functionalities.

3 DEMONSTRATION OVERVIEW

This section presents the setting and plan of our demonstration.

Demonstration setup. A binary version of Rock is available at [1].

Platform. We will demonstrate Rock and its applications by using a single machine powered by 16GB RAM and 8 processors with Intel(R) Xeon(R) Gold 5320 CPU @2.20GH.

Datasets. We will use masked data in the sales domain (see [3]). Additionally, we will utilize two benchmark datasets [15].

System comparison. We will demonstrate the performance of Rock in comparison with SOTA data cleaning systems Raha [16] for error detection, Baran [15] and HoloClean [17] for error correction. Since HoloClean requires logical rules as inputs, we will provide it with the discovered REE^{++}s, but excluding the ML predicates.

ML models. We will embed ML models for *e.g.,* an address normalization model and SKU identification in REE^{++}s as predicates.

Demonstration plan. We will showcase Rock as follows.

Ease of use. Users are invited to play with the interface of Rock, and explore its internal functionalities to understand how these could impact the accuracy/quality of repairs, *e.g.,* the interaction of ER, CR, TD and MI. As shown in Figure 2, Rock provides the following.

(1) Data configuration. Rock can load datasets from various sources, and users may mark a set of attributes of their interest.

(2) Data standardization and ML pre-processing. Users may preprocess the data with built-in rules and pre-trained ML models. They will see (a) the roles of attributes that are automatically predicted,

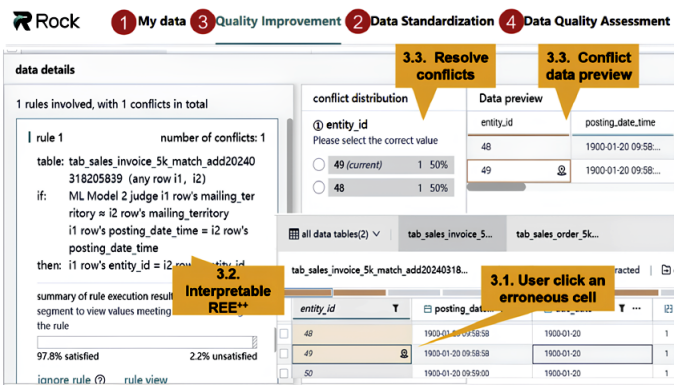

Figure 2: Snapshots of the user panel of Rock

(b) cells with syntactic errors detected by interpretable rules, (c) normalization of cells by (inter-)national standard, (d) enriched attributes by pre-trained ML models and external knowledge graphs, and (e) data standardization by pre-trained ML models.

(3) Data quality enhancement. Users can enhance the quality of their datasets in either the online or offline mode. (a) *Offline mode.* When enough ground truth is accumulated, users may let Rock repair all errors via the chase without user interaction, and return the cleaned datasets at the end. (b) *Online mode.* Users may pick a set of entities of their interest, and interact with Rock to clean the data. As shown in Figure 2, at the user panel, users are shown erroneous (colored) cells detected; the darkness of the colors indicates how certain the cell is erroneous. Users may click on any cell in the tables, and see (a) the REE^{++}s that detect the potential errors; (b) the distribution of conflicts and the recommended value for repair, and (c) a preview of all tuples that have conflicts with the cell. Users may then select/enter the right value for the cell, and Rock extends its ground truth with the input. Rock propagates the corrections to other tuples, incrementally executes the chase, and shows the users with other/new errors. The process proceeds until all errors to entities of users' interest are fixed. The users may opt to trigger incremental rule discovery when the data distribution changes.

(4) Data quality assessment. Users may pick a dataset and view a report with (a) an overall score of its quality, (b) the score of each table in the dataset, (c) error distributions across attributes, tuples and tables, and (d) statistical information, *e.g.*, types/numbers of errors.

Performance. Users are invited to interact with Rock, and see how Rock (a) learns their prior knowledge and discovers top-k REE^{++}s that meet their need, instead of excessive rules, (b) detects and fixes errors in batch and incremental modes, as well as (c) its efficiency, scalability and accuracy, and the impacts of its key parameters on the performance. They will also witness how Rock outperforms SOTA data cleaning systems in both accuracy and efficiency.

Applications. We will walk users through two real-life cases.

(1) Scenario 1: Bank. Reports from banks are required to conform to the directives of regulatory authorities. We will mimic how Rock helps the regulatory reporting system at a bank. Rock discovers REE^{++}s across different tables with arithmetic expressions, and checks consistencies and regulation conformance with the rules. It reduces the regulatory checking from days to hours at the bank.

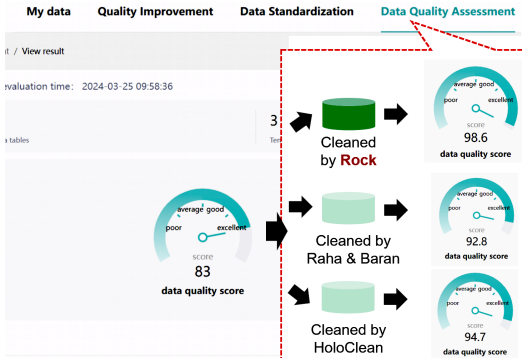

Figure 3: Rock for HR (compared with Raha, Baran **and** HoloClean**)**

(2) Scenario 2: Human resources (HR). We will also show how Rock helps HR manage personnel data. As shown in Figure 3, Rock improves the data quality scores by detecting and correcting errors in their data, *e.g.*, mismatched personnel records, unrecorded changes, outdated salary. It saves hours of HR work each day. Rock outperforms HoloClean and Baran because of its support for unification of logic reasoning and ML predication, and interaction of CR, ER, TD and MI; it is easy to use and ensures its fixes to be certain.

ACKNOWLEDGMENTS

This work was supported by NSFC 62202313, Guangdong Basic and Applied Basic Research Foundation 2022A1515010120, Longhua Science and Technology Innovation Bureau 10162A20220720B12AB12.

REFERENCES

[1] 2023. Rock. http://www.grandhoo.com/en.
[2] Marcelo Arenas, Leopoldo Bertossi, and Jan Chomicki. 1999. Consistent Query Answers in Inconsistent Databases. In *PODS*. 68–79.
[3] Bao et. al. 2024. Rock: Cleaning Data by Embedding ML in Logic Rules. In *SIGMOD (industrial paper)*. ACM.
[4] Exasol. 2020. Exasol Research Finds 58% of Organizations Make Decisions Based on Outdated Data. *https://www.exasol.com/news-exasol-research-finds-organizations-make-decisions-based-on-outdated-data/*.
[5] Wenfei Fan, Hong Gao, Xibei Jia, Jianzhong Li, and Shuai Ma. 2011. Dynamic constraints for record matching. *VLDB J.* 20, 4 (2011), 495–520.
[6] Wenfei Fan, Ling Ge, Ruochun Jin, Ping Lu, and Wenyuan Yu. 2022. Linking Entities across Relations and Graphs. In *ICDE*. IEEE, 634–647.
[7] Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. 2008. Conditional Functional Dependencies for Capturing Data Inconsistencies. *ACM Trans. Database Syst.* 33, 1 (2008), 25:1–25:49.
[8] Wenfei Fan, Ziyan Han, Weilong Ren, Ding Wang Yaoshu Wang, Min Xie, and Mengyi Yan. 2024. Splitting Tuples of Mismatched Entities. *Proc. ACM Manag. Data* (2024).
[9] Wenfei Fan, Ziyan Han, Yaoshu Wang, and Min Xie. 2022. Parallel Rule Discovery from Large Datasets by Sampling. In *SIGMOD*. ACM, 384–398.
[10] Wenfei Fan, Ziyan Han, Yaoshu Wang, and Min Xie. 2023. Discovering Top-k Rules using Subjective and Objective Criteria. *Proc. ACM Manag. Data* (2023).
[11] Wenfei Fan, Chao Tian, Yanghao Wang, and Qiang Yin. 2021. Parallel Discrepancy Detection and Incremental Detection. *PVLDB* 14, 8 (2021), 1351–1364.
[12] Wenfei Fan, Resul Tugay, Yaoshu Wang, Min Xie, and Muhammad Asif Ali. 2023. Learning and Deducing Temporal Orders. *PVLDB* 16, 8 (2023), 1944–1957.
[13] Rachael A Hughes, Jon Heron, Jonathan AC Sterne, and Kate Tilling. 2019. Accounting for missing data in statistical analyses: Multiple imputation is not always the answer. *International journal of epidemiology* 48, 4 (2019), 1294–1304.
[14] Clyde P. Kruskal, Larry Rudolph, and Marc Snir. 1990. A Complexity Theory of Efficient Parallel Algorithms. *Theor. Comput. Sci.* 71, 1 (1990), 95–132.
[15] Mohammad Mahdavi and Z. Abedjan. 2020. Baran: Effective Error Correction via a Unified Context Representation and Transfer Learning. *PVLDB* (2020).
[16] Mohammad Mahdavi, Ziawasch Abedjan, Raul Castro Fernandez, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. 2019. Raha: A Configuration-Free Error Detection System. In *SIGMOD*. 865–882.
[17] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. 2017. HoloClean: Holistic Data Repairs with Probabilistic Inference. *PVLDB* (2017).