IHCS: An Integrated Hybrid Cleaning System

Congcong Ge[†], Yunjun Gao[†♯], Xiaoye Miao[‡], Lu Chen[§], Christian S. Jensen[§], Ziyuan Zhu[†]

[†]College of Computer Science, Zhejiang University, Hangzhou, China
[‡]Center for Data Science, Zhejiang University, Hangzhou, China
[§]Department of Computer Science, Aalborg University, Denmark
[♯]Alibaba–Zhejiang University Joint Institute of Frontier Technologies, Hangzhou, China

[†‡]{gcc, gaoyj, miaoxy, zhu_zy}@zju.edu.cn [§]{luchen, csj}@cs.aau.dk

ABSTRACT

Data cleaning is a prerequisite to subsequent data analysis, and is know to often be time-consuming and labor-intensive. We present IHCS, a hybrid data cleaning system that integrates error detection and repair to contend effectively with multiple error types. In a preprocessing step that precedes the data cleaning, IHCS formats an input dataset to be cleaned, and transforms applicable data quality rules into a unified format. Then, an MLN index structure is formed according to the unified rules, enabling IHCS to handle multiple error types simultaneously. During the cleaning, IHCS first tackles abnormalities through an abnormal group process, and then, it generates multiple data versions based on the MLN index. Finally, IHCS eliminates conflicting values across the multiple versions, and derives the final unified clean data. A visual interface enables cleaning process monitoring and cleaning result analysis.

PVLDB Reference Format:
Congcong Ge, Yunjun Gao, Xiaoye Miao, Lu Chen, Christian S. Jensen, Ziyuan Zhu. IHCS: An Integrated Hybrid Cleaning System. *PVLDB*, 12(12): 1874-1877, 2019.
DOI: https://doi.org/10.14778/3352063.3352088

1. INTRODUCTION

Analyses of dirty data may yield incorrect results that in turn lead to wrong decisions with adverse financial impact on a company [5]. As a consequence, there has been a surge of interest from both industry and academia in data cleaning [2]. Data cleaning often encompasses two stages, i.e., error detection (where dirty data is identified) and error repair (where the data is corrected). Although many approaches exist to detect [7] or repair errors [9], data cleaning remains hard as different methods have different limitations in terms of the errors they address. There are two major reasons for this state of affairs. First, we still lack a unified cleaning model for handling different types of errors. Second, existing approaches often depend too much on the availability of external clean data. Furthermore, existing data cleaning methods tend to separate error detection and error repair, ignoring the connection between them, which adversely affects the cleaning result quality. In addition, existing systems tend to integrate different error detection or repair methods as black boxes [7], making it hard for users to observe occurrences of errors and how they were corrected.

Proceedings of the VLDB Endowment, Vol. 12, No. 12
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3352063.3352088

We address four major challenges: data quality rule multiplicity, limitations in fixable error types, the disconnection between error detection and repair, and the difficulty of analyzing cleaning results. In brief, our goal is to handle different errors including both detection and repair without relying on external clean data, and to enable users to analyse cleaning results in a friendly, visual manner. In the next section, we present the challenges addressed in more detail, using a real-world hospital data for illustration.

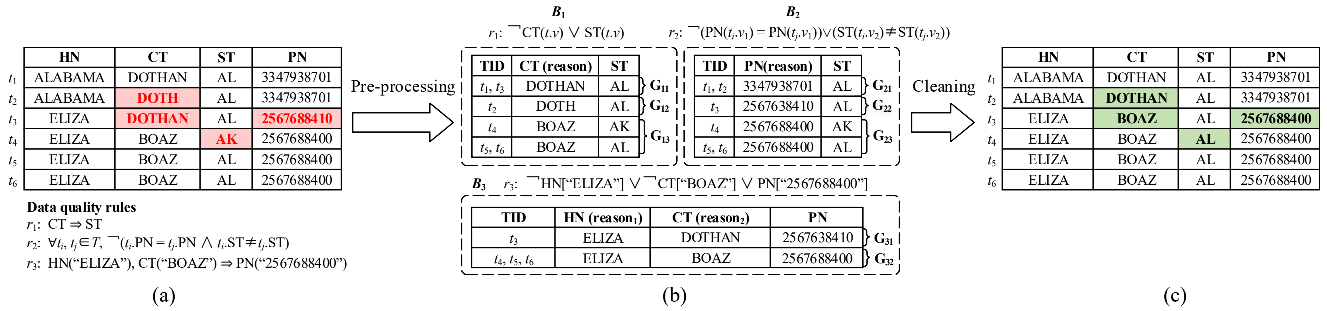
We demonstrate IHCS, an integrated hybrid data cleaning system, to tackle the challenges mentioned above. It consists of two parts: (i) a hybrid cleaning framework that detects and repairs different types of errors; and (ii) a user-friendly interactive interface.

The hybrid cleaning framework accomplishes cleaning in five steps. First, it receives user input (i.e., a dirty dataset with related data quality rules), formats the dataset into a standard form that consists of a set of tuples, and transforms the rules into the unified format of Markov logic network (MLN) rules [4]. Next, it reconstructs the dataset according to the rules, which yields an MLN index structure. The index is used to generate multiple data versions, and to prepare for the subsequent cleaning process. Also, it enables IHCS to handle multiple error types simultaneously. Third, IHCS executes an abnormal group process strategy that eliminates abnormalities in the MLN index. Fourth, IHCS executes a strategy which detects and repairs errors based on the rules within each data version. Finally, it eliminates conflicting values and derives the final unified clean data on top of multi-version data.

The user-friendly interactive interface enables users to input rules and dataset with different formats, track the proceedings of each step of the cleaning process, and interact with the dataset, in order to see the change before and after cleaning from both a tuple-based perspective and an attribute-based perspective. Moreover, it enables evaluation of cleaning results according to multiple evaluation indicators, including runtime, precision, recall, and F1-score. The interface is designed to make the entire cleaning process transparent to users, and it makes it possible for users to further analyze occurrences of errors.

2. MOTIVATION & CHALLENGES

We motivate the problem with a real-world hospital dataset and present four challenges addressed by IHCS, namely data quality rule multiplicity, limitations in fixable error types, the disconnec-



	HN	CT	ST	PN
t_1	ALABAMA	DOTHAN	AL	3347938701
t_2	ALABAMA	DOTH	AL	3347938701
t_3	ELIZA	DOTHAN	AL	2567688410
t_4	ELIZA	BOAZ	AK	2567688400
t_5	ELIZA	BOAZ	AL	2567688400
t_6	ELIZA	BOAZ	AL	2567688400

Data quality rules

r_1: CT ⇒ ST

r_2: $\forall t_i, t_j \in T$, $\neg(t_i.\text{PN} = t_j.\text{PN} \wedge t_i.\text{ST} \neq t_j.\text{ST})$

r_3: HN("ELIZA"), CT("BOAZ") ⇒ PN("2567688400")

B_1

r_1: $\neg\text{CT}(t.v) \vee \text{ST}(t.v)$

TID	CT (reason)	ST	
t_1, t_3	DOTHAN	AL	G_{11}
t_2	DOTH	AL	G_{12}
t_4	BOAZ	AK	G_{13}
t_5, t_6	BOAZ	AL	

B_2

r_2: $\neg(\text{PN}(t_i.v_1) = \text{PN}(t_j.v_1)) \vee (\text{ST}(t_i.v_2) \neq \text{ST}(t_j.v_2))$

TID	PN(reason)	ST	
t_1, t_2	3347938701	AL	G_{21}
t_3	2567638410	AL	G_{22}
t_4	2567688400	AK	G_{23}
t_5, t_6	2567688400	AL	

B_3 r_3: $\neg$HN["ELIZA"] $\vee \neg$CT["BOAZ"] $\vee$ PN["2567688400"]

TID	HN (reason$_1$)	CT (reason$_2$)	PN	
t_3	ELIZA	DOTHAN	2567638410	G_{31}
t_4, t_5, t_6	ELIZA	BOAZ	2567688400	G_{32}

	HN	CT	ST	PN
t_1	ALABAMA	DOTHAN	AL	3347938701
t_2	ALABAMA	DOTHAN	AL	3347938701
t_3	ELIZA	BOAZ	AL	2567688400
t_4	ELIZA	BOAZ	AL	2567688400
t_5	ELIZA	BOAZ	AL	2567688400
t_6	ELIZA	BOAZ	AL	2567688400

Figure 1: IHCS Workflow Diagram Using a Motivating Example

tion between error detection and repair, and the difficulty of analyzing cleaning results.

Example 2.1. Figure 1(a) depicts a group of tuples from a real-world hospital dataset T complying with three data quality rules. The tuples have four attributes, including hospital name (HN), city (CT), state (ST), and phone number (PN). We utilize the notation t_i and t_j $(i \neq j)$ to represent tuples. Rule r_1 states that a city determines a state, rule r_2 states that two hospitals located in different states have different phone numbers, and rule r_3 states that the hospital named "ELIZA" and located in city "BOAZ" has phone number "2567688400". Errors in the dataset, including *substitution errors*, *typos*, and *values that violate data quality rules*, are highlighted in red. Substitution errors may occur when users select an incorrect value from an attribute domain that contains multiple values. For instance, t_3.[CT] having value "DOTHAN" is a substitution error, and the correct value is "BOAZ". Typos, also called misprints, are artifacts of the typing process. For example, t_2.[CT] having value "DOTH" is a typo, and the correct value is "DOTHAN". The tuple pairs (t_4, t_5) and (t_4, t_6) violate rule r_1.

First, the data quality rules in the example occur in different formats. It is hard to clean errors that violate rules expressed in different formats. This leads to data quality rule multiplicity.

Second, errors may appear anywhere in the dataset and may include substitution errors, typos, and values that violate rules. However, existing methods are not able to contend with all the error types mentioned above. Qualitative techniques [3, 7] find that tuples t_4, t_5, and t_6 represent a violation on the attribute ST w.r.t. r_1. Hence, according to the principle of minimality, these techniques replace the value "AK" with "AL" in t_4, but they fail to repair the values of attributes CT and PN of t_3. In addition, the value of attribute CT of t_2 cannot be repaired since it does not violate any rule. In contrast, quantitative techniques [8, 9] do not consider rules but instead ensure that the cleaning result conforms to statistical characteristics. Nonetheless, they rely for their functioning on the availability of sufficient external clean data to train a reliable model; otherwise, the result of the cleaning is of low quality.

Third, existing cleaning methods either focus on error detection or on error repair [9], while disregarding the connections between them. This may decrease the cleaning result quality.

Last but not least, existing systems tend to integrate different error detection or repair methods as black boxes, making it hard for users to analyze cleaning results and thus understand which values are wrong and why.

3. SYSTEM OVERVIEW

The IHCS architecture is shown in Figure 2. It is composed of two parts: (i) a hybrid cleaning framework; and (ii) a user-friendly interactive interface. The hybrid framework aims to detect and repair different types of errors. The interface makes the cleaning process transparent to users, and provides an assessment of the cleaning results for helping users to analyze the occurrence of errors.

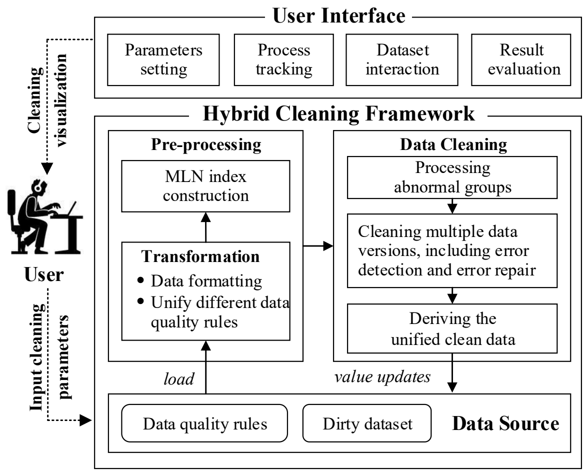


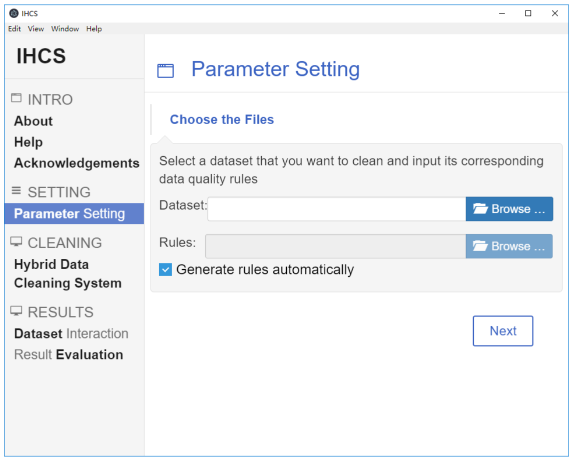
Figure 2: IHCS Architecture

First, we overview each module in the hybrid cleaning framework, and illustrate their functionalities using the motivating example. The framework features a pipelined modular architecture consisting of two phases, i.e., *pre-processing* and *data cleaning*, which have two and three modules, respectively. A detailed description of the hybrid cleaning framework can be found in [6].

Transformation. This module implements a method that automatically transforms the input dataset into a standard form with a set of tuples. Further, it converts different data quality rules into MLN rules, which are first-order clauses, to unify different types of rules. An MLN rule has the form $l_1 \vee l_2 \vee ... \vee l_n$, where l_i is a literal, $i = 1, \cdots, n$. A literal is any expression that contains a predicate symbol applied to a variable or a constant, e.g., CT$(t.v)$, HN("ELIZA"). Each data quality rule can be considered as having two parts, i.e., a *reason* part and a *result* part, and the result part is a logical consequence of the reason part. There is no the same reason to derive multiple different results. In MLN rules, we treat the last predicate as the result part, and the other predicates as the reason part. For rule r_1, CT$(t.v)$ is the reason, and ST$(t.v)$ is the result.

Since the dataset in Figure 1(a) is already standardized, we only need to transform the related rules into MLN rules. The transformed rules are depicted in Figure 1(b).

MLN index construction. In this module, we develop an MLN index, which is used to generate multiple data versions and to handle multiple error types simultaneously. The first layer consists of a set of *blocks*, each of which has a set of *groups* in the second layer. One block corresponds to one MLN rule, and each group consists of a set of *pieces of data* (γs for short) with the same reason part, and each γ with regard to multiple values of tuples involving one



Choose the Files

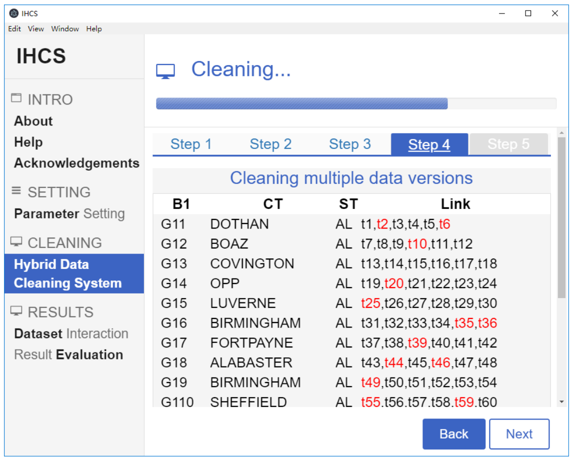
Select a dataset that you want to clean and input its corresponding data quality rules

Dataset:

Rules:

☑ Generate rules automatically

Figure 3: Parameter Setting Visualization



Cleaning multiple data versions

B1	CT	ST	Link
G11	DOTHAN	AL	t1,t2,t3,t4,t5,t6
G12	BOAZ	AL	t7,t8,t9,t10,t11,t12
G13	COVINGTON	AL	t13,t14,t15,t16,t17,t18
G14	OPP	AL	t19,t20,t21,t22,t23,t24
G15	LUVERNE	AL	t25,t26,t27,t28,t29,t30
G16	BIRMINGHAM	AL	t31,t32,t33,t34,t35,t36
G17	FORTPAYNE	AL	t37,t38,t39,t40,t41,t42
G18	ALABASTER	AL	t43,t44,t45,t46,t47,t48
G19	BIRMINGHAM	AL	t49,t50,t51,t52,t53,t54
G110	SHEFFIELD	AL	t55.t56.t57.t58.t59.t60

Figure 4: Cleaning Processing Visualization

rule. For example, G_{13} includes two pieces of data, denoted as γ_1 ={CT:BOAZ, ST:AK} and γ_2 ={CT:BOAZ, ST:AL}. We call the collection of γs in a block a *data version*. Thus, there are *multiple data versions*, each of which comes from a different block.

The MLN index of the sample dataset is shown in Figure 1(b). There are three blocks B_1, B_2, and B_3 related to three rules r_1, r_2, and r_3, respectively. They have 3, 3, and 2 groups, respectively.

Processing abnormal groups. Based on an MLN index, for a tuple with errors (including substitutions and typos) in the reason part of a rule, the corresponding piece of data (γ for short) might erroneously forms or belongs to a group. In that case, we call that group *an abnormal group*. In this module, an *abnormal group process* strategy first identifies abnormal groups and then merges them with corresponding normal groups. Each abnormal group in a block is merged with its most similar normal group.

In the MLN index shown in Figure 1(b), groups G_{12}, G_{22}, and G_{31} are abnormal groups. G_{12} is merged with G_{11}, G_{22} is merged with G_{23}, and G_{31} is merged with G_{32}.

Cleaning multiple data versions. In the MLN index, the data pieces in a group share the same value(s) on the reason part of the corresponding rule. Ideally, if the data is clean, one group contains exactly one piece of data, meaning that the same values of the reason part cannot derive different values on the result part. When one group contains multiple pieces of data, dirty values exist. Therefore, we include a strategy that cleans dirty values within each group. The strategy determines which piece of data contained in a group is clean using the concept of *reliability score*. The higher the reliability score is, the more likely it is that a piece of data is clean. The clean data then replaces the remaining dirty data, so that each group eventually has exactly one piece of data.

In our motivating example, the first clean data version contains {CT: DOTHAN, ST: AL} in G_{11} and {CT: BOAZ, ST: AL} in G_{13}. The second clean data version incorporates {PN: 3347938701, ST: AL} in G_{21} and {PN: 2567688400, ST: AL} in G_{23}. The third one contains {HN: ELIZA, CT: BOAZ, PN: 2567688400} in G_{32}.

Deriving the unified clean data. This module forms the final clean data on top of the multi-version data, in which values that conflict across different data versions are resolved. This module provides the second opportunity to clean errors, including substitutions and typos, which are not (or are incorrectly) repaired in the previous cleaning step. First, all conflict values in each tuple are detected. Then, IHCS regenerates each value of the tuple containing conflicts. Since there are multiple different conflict values for each tuple, we get a series of regenerated results for each tuple. The novel concept of *fusion score* is introduced to calculate the Markov weight of each regenerated result, so as to get the most likely clean version as the final result for each tuple.

Take tuple t_3 in Figure 1(a) as an example. After finishing this cleaning step, t_3.[CT] has two different values (i.e., "DOTHAN" and "BOAZ") from the different versions. Thus, there is a conflict for attribute CT of t_3, which should be eliminated to get the final clean t_3. In addition, although {CT: DOTHAN, ST: AL} conforms to rule r_1, there might still exist errors w.r.t t_3 if it is erroneously classified into a group and thereby could not be repaired correctly in the previous cleaning step. After executing this module, the final regenerated result of t_3, i.e., {HN: ELIZA, CT: BOAZ, ST: AL, PN: 2567688400}, is obtained. The final cleaning result is illustrated in Figure 1(c).

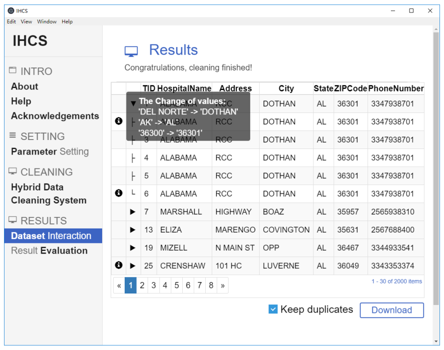
Next, we cover each module in the user interface.

Parameters setting. In this module, users are required to input a dirty dataset and its corresponding data quality rules. We accept datasets with different formats (e.g., .csv, .txt, .json) and rules using different logical languages (i.e., first-order logic and clausal form). Further, since users may not have sufficient knowledge associated with the dataset, it may be difficult for them to state rules. We integrate an effective rule generation algorithm [1] that automatically generates a series of possible rules based on the dataset.

Process tracking. In this module, users can track the cleaning process through two components, i.e., a progress bar and a progress tab. The progress bar is used to show the completion extent of the current cleaning task. The progress tab offers more detail. It reports the step in which the current task is executed (corresponding to the five cleaning steps included in the framework) and the results produced by each step.

Dataset interaction. This module enables users to view modifications, including value corrections and tuple deduplication, between the dirty dataset and the final cleaned result. For value correction, users can see changes from two perspectives: (i) a tuple-based perspective and (ii) an attribute-based perspective. When using the tuple-based perspective, the module marks tuples that contain fixes and shows the values before and after cleaning. When using the attribute-based perspective, the module enables users to view the values of each attribute before and after cleaning. For deduplication, the module aggregates duplicate tuples, and thus, users are provided information about how many tuples are in each aggregation. Based on this, users can choose whether to keep duplicates when saving the cleaned dataset.

Result analysis. This module enables users to evaluate the runtime and accuracy of the cleaning result. It also records histori-



Results

Congratrulations, cleaning finished!

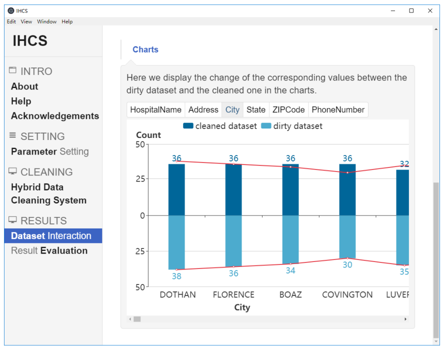
		TID	HospitalName	Address	City	State	ZIPCode	PhoneNumber
	▼	1	ALABAMA	RCC	DOTHAN	AL	36301	3347938701
●	├	2	ALABAMA	RCC	DOTHAN	AL	36301	3347938701
	├	3	ALABAMA	RCC	DOTHAN	AL	36301	3347938701
	├	4	ALABAMA	RCC	DOTHAN	AL	36301	3347938701
	├	5	ALABAMA	RCC	DOTHAN	AL	36301	3347938701
●	└	6	ALABAMA	RCC	DOTHAN	AL	36301	3347938701
	▶	7	MARSHALL	HIGHWAY	BOAZ	AL	35957	2565938310
	▶	13	ELIZA	MARENGO	COVINGTON	AL	35631	2567688400
	▶	19	MIZELL	N MAIN ST	OPP	AL	36467	3344933541
●	▶	25	CRENSHAW	101 HC	LUVERNE	AL	36049	3343353374

The Change of values:
'DEL NORTE' -> 'DOTHAN'
'AK' -> 'AL'
'36300' -> '36301'

« 1 2 3 4 5 6 7 8 » 1 - 30 of 2000 items

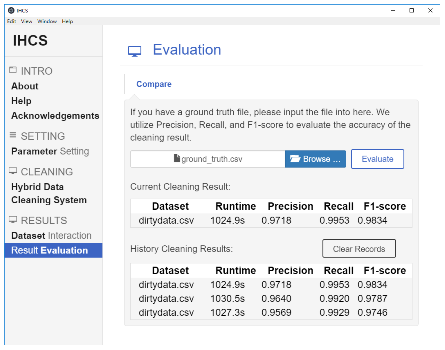
✓ Keep duplicates

(a) Tuple-based perspective



Here we display the change of the corresponding values between the dirty dataset and the cleaned one in the charts.

(b) Attribute-based perspective



Evaluation

Compare

If you have a ground truth file, please input the file into here. We utilize Precision, Recall, and F1-score to evaluate the accuracy of the cleaning result.

ground_truth.csv

Current Cleaning Result:

Dataset	Runtime	Precision	Recall	F1-score
dirtydata.csv	1024.9s	0.9718	0.9953	0.9834

History Cleaning Results:

Dataset	Runtime	Precision	Recall	F1-score
dirtydata.csv	1024.9s	0.9718	0.9953	0.9834
dirtydata.csv	1030.5s	0.9640	0.9920	0.9787
dirtydata.csv	1027.3s	0.9569	0.9929	0.9746

(c) Result evaluation

Figure 5: Cleaning Result Analysis Visualization

cal evaluation results, which reflect the cleaning effect of IHCS on datasets with different source or scale.

4. DEMONSTRATION

The IHCS system is built as a cross-platform desktop application. The basic interface is shown in Figure 3. The left navigation bar illustrates three main parts of the demo: the setting part, the cleaning part, and the results part. We demonstrate IHCS using a real-world hospital dataset (https://data.medicare.gov/data).

First, a user inputs a dirty dataset with corresponding data quality rules in the setting part as depicted in Figure 3. IHCS accepts dirty datasets with different formats. Since the target group of IHCS includes both domain experts and ordinary users, IHCS provides two options for giving rules: one is to enter rules manually (for experts), and the other is to generate rules automatically (for ordinary users). When the user selects the checkbox at the bottom of the page, the system automatically generates rules, and rules can no longer be uploaded manually.

After the parameters are prepared, the user can click the "Next" button to initiate the cleaning. Meanwhile, the system proceeds to the cleaning part. The user can track the progress of the cleaning through the progress bar and the progress tabs. As shown in Figure 4, the current cleaning progress is in the fourth step, i.e., *cleaning multiple data versions*. The result produced at this step is shown below the tab, and it displays a set of clean data versions with their corresponding tuples. After finishing this step, the tuples with changes in attribute values are marked in red. Further, the user can switch among tabs to review the results generated in each step that has been executed. Unexecuted steps are not able to be clicked.

When the cleaning process is completed, the system proceeds to the results part that consists of two subparts, i.e., dataset interaction and result evaluation. In the dataset interaction part, IHCS enables the user to see the change to the dataset from tuple and attribute perspectives. In the tuple-based perspective, as depicted in Figure 5(a), the cleaned dataset is organized as a set of unique tuples. All modified tuples are marked with black dots to the left. When the user hovers over a dot, the modifications to each attribute value of the tuple are shown. Detected duplicate tuples are hidden by default, and the user can click on each tuple to see how many tuples have the same values as it does. Also, before the user clicks the "Download" button to save the cleaned result, it is possible to decide whether or not to keep duplicates by selecting the checkbox at the bottom of this page. Using the attribute-based perspective, IHCS displays the changes to the value of each attribute before and after the cleaning in charts, as shown in Figure 5(b). The user can switch between charts by clicking on the attribute name. The user can infer where errors are likely to occur based on changes to the number of attribute values.

In the result evaluation part, as depicted in Figure 5(c), IHCS evaluates the current cleaning result according to multiple evaluation indicators, including runtime and accuracy (e.g., precision, recall, and F1-score). The runtime is recorded after cleaning, while the accuracy measurement requires the user to provide a ground truth file. Moreover, IHCS allows the user to review historical cleaning result evaluations. As shown in Figure 5(c), we record three evaluations of the hospital dataset, and the results show that IHCS can achieve high cleaning accuracy.

5. ACKNOWLEDGMENTS

This work was supported in part by the National Key R&D Program of China under Grant No. 2018YFB1004003, the NSFC under Grants No. 61522208, the NSFC-Zhejiang Joint Fund under Grant No. U1609217, the ZJU-Hikvision Joint Project, and the Fundamental Research Funds for the Central Universities. Yunjun Gao is the corresponding author of the work.

6. REFERENCES

[1] F. Chiang and R. J. Miller. Discovering data quality rules. *PVLDB*, 1(1):1166–1177, 2008.

[2] X. Chu, I. F. Ilyas, S. Krishnan, and J. Wang. Data cleaning: Overview and emerging challenges. In *SIGMOD*, pages 2201–2206, 2016.

[3] M. Dallachiesa, A. Ebaid, A. Eldawy, A. K. Elmagarmid, I. F. Ilyas, M. Ouzzani, and N. Tang. NADEEF: A commodity data cleaning system. In *SIGMOD*, pages 541–552, 2013.

[4] P. M. Domingos and D. Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2009.

[5] W. W. Eckerson. Data quality and the bottom line: Achieving business success through a commitment to high quality data. *The Data Warehousing Institute*, pages 1–36, 2002.

[6] Y. Gao, C. Ge, X. Miao, H. Wang, B. Yao, and Q. Li. A hybrid data cleaning framework using markov logic networks. *arXiv preprint arXiv1903.05826*, 2019.

[7] Z. Khayyat, I. F. Ilyas, A. Jindal, S. Madden, M. Ouzzani, P. Papotti, J. Quiané-Ruiz, N. Tang, and S. Yin. BigDansing: A system for big data cleansing. In *SIGMOD*, pages 1215–1230, 2015.

[8] S. Krishnan, J. Wang, E. Wu, M. J. Franklin, and K. Goldberg. Activeclean: Interactive data cleaning for statistical modeling. *PVLDB*, 9(12):948–959, 2016.

[9] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré. Holoclean: Holistic data repairs with probabilistic inference. *PVLDB*, 10(11):1190–1201, 2017.