

# EasyDR: A Human-in-the-loop Error Detection&Repair Platform for Holistic Table Cleaning

Yihai Xi  
Beijing Jiaotong University  
Beijing, China  
xiyihai@bjtu.edu.cn

Yiyi Zhang  
Beijing Jiaotong University  
Beijing, China  
Zhangyiyi@bjtu.edu.cn

Ning Wang  
Beijing Jiaotong University  
Beijing, China  
nwang@bjtu.edu.cn

Zilong Wang  
Beijing Jiaotong University  
Beijing, China  
21120413@bjtu.edu.cn

Yue Wang  
Beijing Jiaotong University  
Beijing, China  
21125254@bjtu.edu.cn

Xinyu Chen  
Beijing Jiaotong University  
Beijing, China  
ChenXinyu@bjtu.edu.cn

Zhihong Xu  
Beijing Jiaotong University  
Beijing, China  
21120430@bjtu.edu.cn

## ABSTRACT

Many tables on the web suffer from multi-level and multi-type quality problems, but existing cleaning systems cannot provide a comprehensive quality improvement for them. Most of these systems are designed for solving a specific type of error, so that we need to resort to a number of different cleaning tools (one per error type) to get a high quality table. In this demonstration, we propose a human-in-the-loop cleaning platform EasyDR for detecting and repairing multi-level&multi-type errors in tables. The attendees will experience the following features of EasyDR: 1) Holistic error detection&repair. Users are able to perform a holistic table cleaning in EasyDR where machine algorithms are responsible for error detection while human intelligence is leveraged for error repairing. 2) Human-in-the-loop table cleaning. EasyDR performs an all-round quality diagnosis for the table, and automatically generates crowdsourcing cleaning tasks for the detected errors. To simplify cleaning tasks for crowdsourcing workers, EasyDR provides two task optimization techniques including domain-aware table summarization and difficulty-aware task order optimization. 3) Customizable cleaning mode. EasyDR provides a declarative language for users to customize cleaning tasks flexibly, e.g., selecting target errors, restricting the cleaning scope, defining the cooperation mode for machine and crowd.

## PVLDB Reference Format:

Yihai Xi, Ning Wang, Xinyu Chen, Yiyi Zhang, Zilong Wang, Zhihong Xu, and Yue Wang. EasyDR: A Human-in-the-loop Error Detection&Repair Platform for Holistic Table Cleaning. PVLDB, 15(12): 3578 - 3581, 2022. doi:10.14778/3554821.3554848

\*Ning Wang is the corresponding author.

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit <https://creativecommons.org/licenses/by-nc-nd/4.0/> to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing [info@vldb.org](mailto:info@vldb.org). Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.

Proceedings of the VLDB Endowment, Vol. 15, No. 12 ISSN 2150-8097. doi:10.14778/3554821.3554848

## 1 INTRODUCTION

Relational tables on the web are rich sources of structured data for collecting and analyzing data, but there exist various data quality problems due to the lack of strict constraints. Generally, there are two levels of quality problems in a web table, i.e., schema-level errors and instance-level errors. Schema-level errors exist in table schema such as table title, subject column (one entity column that can determine the values of other columns), and column names. Instance-level errors lie in tuples where each one describes some facts of an instance in the subject column. To obtain a high quality web table, we need to perform a holistic cleaning task including error detection and data repair for the above two levels of errors.

Existing cleaning systems are mainly divided into three categories. Machine-based systems such as HoloClean [6] and CleanM [3] rely on integrity constraints or external data sources to clean errors. Human-based systems like CrowdFill [5] leverage human intelligence to do cleaning tasks. Hybrid human-machine systems such as KATARA [2], Corleone [4] and HOP [1] combine human intelligence and machine algorithms for cleaning. Although there are lots of cleaning systems, they still have three limitations to perform a holistic cleaning for tables.

First, existing cleaning systems are not scalable for multi-level&multi-type errors. Most of them are designed for solving a specific type of error and cannot settle other types of quality problems in tables. Thus, to get a high quality table, we need to resort to a number of different cleaning tools - one per error type. In addition, existing systems usually focus on instance-level errors while ignoring schema-level ones. In fact, the quality of a table depends on the quality of both schema and instance in the table, and the complete and correct schema information can guide the detecting and repairing for instance-level errors. Thus, it is important to do the holistic table cleaning for multi-level&multi-type errors.

Second, human and machine are not well combined to provide a holistic cleaning solution including error detection and data repair. Some errors are easy to detect but may be hard to repair by machine, especially for the multi-modal tables consisting of words and

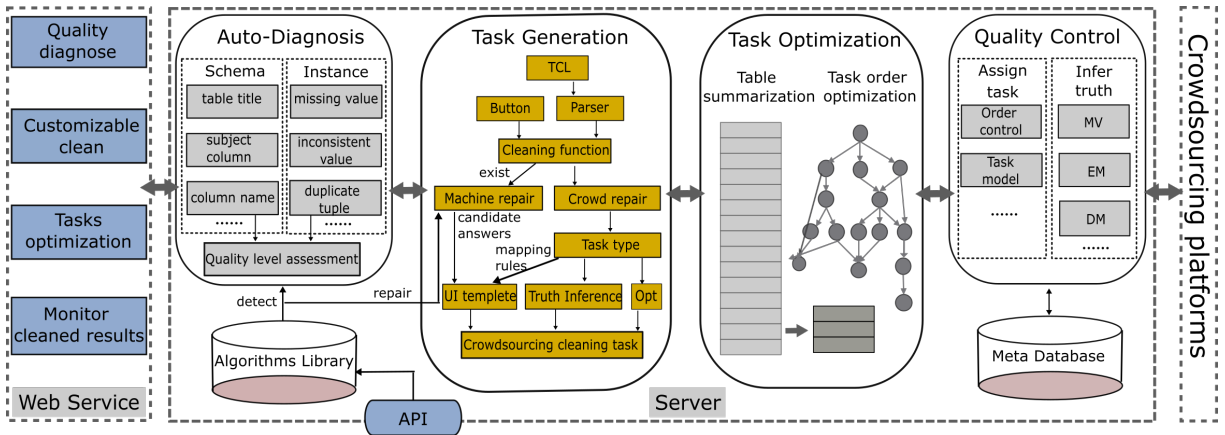


Figure 1: The Architecture of EasyDR

images. In fact, machine-based methods are good at error detection while humans are good at data repair especially for machine-hard problems such as image recognition. However, it is difficult to guarantee the cleaning task quality by crowdsourcing platform for most workers are not experts in data cleaning.

Third, it is inconvenient for users to leverage crowdsourcing to do cleaning tasks in existing systems. To design a human intelligence task (HIT) for cleaning one error in the table, the user needs to think about various aspects including setting task type, designing question content, and arranging table presentation.

To address these challenges, we propose a human-in-the-loop cleaning platform EasyDR for detecting and repairing multi-level and multi-type quality errors in tables. It has the following features:

**Holistic and scalable error detection&repair.** EasyDR is designed to perform a comprehensive quality improvement for tables. First, EasyDR provides a unified error detection and repair solution for tables. In EasyDR, machine algorithms are mainly responsible for error detection while human intelligence is leveraged for repairing machine-hard errors. Second, EasyDR is scalable to settle multi-level&multi-type quality problems. It provides plug-and-play service for new detection algorithms and settles emerging quality problems by embedding new cleaning tasks into our platform.

**Human-in-the-loop table cleaning.** Given a pending table, EasyDR will perform an all-round quality diagnosis, and then automatically generate cleaning tasks for machine and crowd. To improve the cleaning quality by crowd, we propose two task optimization techniques including domain-aware table summarization and difficulty-aware task order optimization to simplify cleaning tasks and help crowd workers clean errors effectively and efficiently.

**Customizable cleaning mode.** To help users easily settle various quality problems in tables, we design a Table Cleaning Language (TCL) in EasyDR to customize the cleaning tasks flexibly. We also provide a click&clean approach, where users just need to click the mouse on a detected type of error and EasyDR automatically starts the cleaning work. Both of the TCL and click&clean approach are user-friendly, because users only need to decide on what type of error to clean rather than how to design specific cleaning tasks.

## 2 SYSTEM OVERVIEW

EasyDR is composed of two main components: web service and server, as shown in Fig.1.

**Web service.** To help users perform a holistic cleaning for tables, we provide four main functions in the web interface: 1) demonstrating the result of quality diagnosis including the proportion of each type of error in the pending table; 2) guiding users to do cleaning tasks for detected errors - one click per error type; 3) providing task optimization options for users to optimize the generated HITs; 4) presenting repaired values in the original table in real time.

**Server.** An end-to-end detection and repair pipeline is supported by four modules:

(1) Auto-Diagnosis module performs a quality diagnosis for the pending table including an overall quality assessment and the detection of different types of errors. EasyDR integrates some popular detection algorithms into the Algorithm Library. At present, it supports the detection of schema-level errors including: 1) inappropriate table title, 2) missing subject column, 3) missing/inappropriate column name, and instance-level errors including: 1) missing attribute value, 2) inconsistent attribute value (including outliers), 3) duplicate tuples. For scalability, it also provides an API for new detecting and repairing algorithms to join EasyDR.

(2) Task Generation module is designed to enable users to trigger a cleaning task through a TCL or a click&clean button. We design a task-centric cleaning approach where a list of cleaning tasks are identified by EasyDR for mapping each task to a single function. When a function is called by a TCL or a click&clean button, EasyDR will execute the function to generate corresponding cleaning tasks. If there exists the repairing algorithm for one type of error, EasyDR can use the algorithm to repair part of the errors or provide candidate answers for crowd workers. For the machine-hard errors left, EasyDR will generate HITs based on a set of mapping rules about the elements of the HIT such as UI template.

(3) Task Optimization module provides two crowdsourcing task optimization techniques. For schema-level errors, it performs domain-aware table summarization that presents notable tuples with broad coverage and high diversity of domains to help workers understand

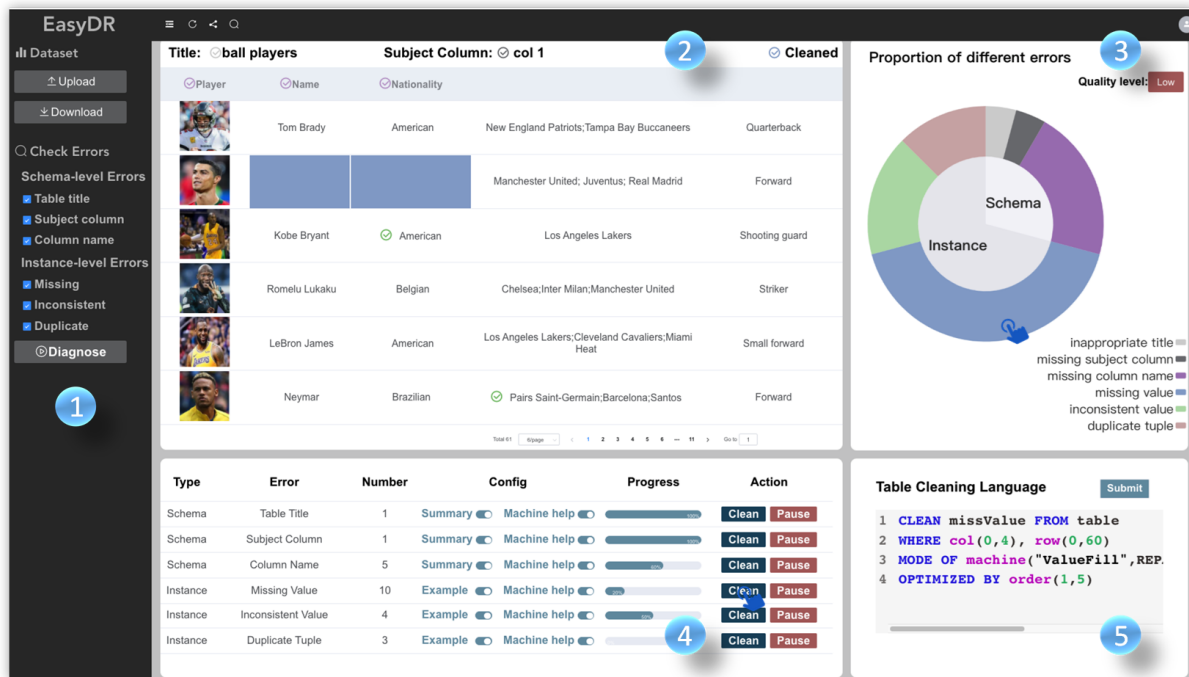


Figure 2: A Screenshot of EasyDR

table topic and repair the schema-level errors easily. For instance-level errors, it recommends optimized task execution order from easy to difficult to help workers learn experience from easy tasks and be more experienced for the similar but more difficult tasks.

(4) Quality Control module takes on task assignment and truth inference for improving the overall result accuracy. HITs are assigned to workers based on the task model and the optimized execution order of tasks. Several truth inference methods are provided to deal with different types of cleaning tasks. In addition, we design a novel decision-making (DM) model to deal with special tasks with uncertain number of answers such as a mix of single-choice and completion question, which makes our platform more scalable for emerging cleaning tasks.

### 3 DEMONSTRATION OVERVIEW

Fig.2 is a screenshot of the front-end of EasyDR. The user can observe the detection and repair pipeline with the following steps:

**Step 1 (Holistic quality diagnose.)** First, the user can upload a table by “Upload” button (see Fig.2-1). We have prepared a low quality table containing 60 ball players. Then, EasyDR performs an all-round quality diagnosis for the table after clicking “Diagnose” button. EasyDR also provides a “Check Errors” panel for the user to select the error types to be detected.

After the diagnosis, the overall quality level of the table and the proportion of different errors are presented by a pie chart (see Fig.2-3). In EasyDR, we design four levels as Top, High, Mid, Low to evaluate the quality of table according to error severity in the table, which takes into account both the number of errors and the




error type with a weighted score based on its importance in the table. If the user wants to further observe one specific error type, she just needs to click the error in the pie chart to make all errors of this type exposed in the table. For example, after clicking the error of missing value in the pie chart, all missing attribute values in the table are exposed in blue (see Fig.2-2).

**Step 2 (Customizable table cleaning.)** EasyDR lists all detected errors at the bottom left of interface (see Fig.2-4). The user can click the “Clean” button for the type of error that she wants to clean, then EasyDR will automatically call the cleaning function to generate cleaning tasks with default settings. Users can set some simple parameters for the cleaning tasks. Button “Machine help” is used to control whether our platform uses repairing algorithms to clean errors directly or provides candidate answers for crowds. Button “Summary” controls whether our platform uses the table summarization technique to optimize the cleaning tasks for schema-level errors. Button “Example” controls whether our platform optimizes the execution order of instance-level cleaning tasks and provides examples for crowd workers. However, if the user wants to customize the cleaning task, she can express cleaning needs by TCL from the language entry (see Fig.2-5).

**Step 3 (Crowdsourcing tasks delivering.)** Fig.3 shows an auto-generated cleaning task for filling in missing column name. The candidate answers come from the machine repairing algorithm, and our platform also provides a blank box for workers to fill in the answer if she does not agree with those candidates. The task has been optimized by our summarization technique with default setting (summary size = 3). Obviously, the tuples in the task are

ID:CN-002

Question: Please, fill in the suitable label of the highlighted column.

	Michael Jordan	American	Chicago Bulls	Shooting Guard
	Lionel Messi	Argentina	Paris Saint-Germain	Forward
	Tom Brady	American	Tampa Bay Buccaneers	Quarterback

Write Your Answer:

Candidate Answer: 1. Person 2. Player 3. Basketball player

Name

Figure 3: An Example of Auto-generated Cleaning Task

filled with notable entities and can cover significant domains in the table, which are more helpful than random tuples for inferring the missing column name. The attendees will be encouraged to finish those tasks as crowd workers.

**Step 4 (Results monitoring.)** When a cleaning task is finished, the repaired values will be shown in the original table with a check mark on the left, and the color of the check mark is corresponding to the color of the error in the pie chart. For example, after finishing the completion task in Fig.3, the repaired column name “Name” is shown in the table (see Fig.2-2) with a check mark in purple.

## 4 CORE TECHNIQUES

**1 Table cleaning language.** The syntax of table cleaning language is shown in the box. The symbols `[]` and `|` denote optional elements and option between elements, respectively. Operator **CLEAN** selects the target error type to be cleaned, **FROM** determines the target table, **WHERE** restricts the cleaning scope, **MODE OF** defines cleaning tasks for machine and crowd, and **OPTIMIZED BY** chooses the optimization technique.

```

CLEAN <ERROR> FROM <TABLE>
[WHERE <SCOPECLAUSE>]
[MODE OF <TASKCLAUSE>]
[OPTIMIZED BY <OPTCLAUSE>]
<SCOPECLAUSE> ::= col(<Beg>, <End>)|row(<Beg>, <End>)|
                (col(<Beg>, <End>), row(<Beg>, <End>))
<TASKCLAUSE> ::= <CROWDTASK>|<HYBRIDTASK>
<CROWDTASK> = crowd(<Type> [, <TruthInfer>, <Require>])
<HYBRIDTASK> = machine(<Algorithm>, REPAIR|CANDIDATE),
                <CROWDTASK>
<OPTCLAUSE> ::= summary(<Size>)|order(<Space>, <Latency>)

```

Specifically, `<SCOPECLAUSE>` consists of two functions `col()` and `row()`, and fields `<Beg>` and `<End>` correspond to the begin and end index number of columns or rows. If the clause is not specified, the cleaning task will be performed on the overall table; otherwise, it is performed on the specified column(s) and row(s). `<TASKCLAUSE>` consists of two cleaning modes, i.e., `<CROWDTASK>` refers to cleaning errors by crowdsourcing while `<HYBRIDTASK>` refers to repairing by the cooperation of machine and crowd. Function `crowd()` comprises of `<Type>` for the type of HIT (e.g., choice task), `<TruthInfer>` for the truth inference method, and `<Require>` for the number of answers required. For `<HYBRIDTASK>`,

the `<Algorithm>` is provided for either repairing the errors (`<Repair>`) or giving candidate answers (`<Candidate>`). `<OPTCLAUSE>` consists of two optimization functions, where `summary()` summarizes the table by the value of `<Size>` and `order()` optimizes an execution order of tasks under constraints `<Space>` and `<Latency>`.

**2 Domain-aware table summarization.** Table summarization [7] aims to provide a concise and informative table overview and assist workers to clean the table. In EasyDR, we capture the properties of notability and domain to provide a high quality table summary. First, we propose a fine-grained ranking of entities based on their significance in the real world. We observe that notable entities get stronger relationship with others in the table. Consequently, we map entities in the table to knowledge bases and construct an entity link graph to calculate the notability of each entity. Second, we derive clusters of similar entities based on domain loss, which measures the coverage and significance of domains in a summary. Then, clusters are adjusted to improve entity gain reflecting the domain diversity of the summarization. Finally, we select entities in the centroid of each cluster to form a broad-coverage and high-diversity summary consisting of notable entities.

**3 Difficulty-aware task order optimization.** The difficulty of cleaning tasks is influenced by many factors such as task type and worker expertise. In EasyDR, we seek a more general evaluation method based on the task notability while being independent of the worker model, for most crowdsourcing platforms cannot provide the worker information for the consideration of privacy. With the task difficulty, we optimize the execution order of cleaning tasks from easy to difficult, and let the cleaned tasks be examples for the succeeding tasks to make them understandable. Specifically, we construct a task order graph to represent the execution order of tasks, where each vertex denotes one task and each direct edge represents the execution sequence of two tasks. However, the topological execution order of tasks will introduce the latency and tasks should take up little space due to the page size of computers or phones. To meet the constraints, we propose difficulty loss to evaluate the importance of each task sequence in the graph for reducing task difficulty. Then we use a heuristic algorithm to find the optimized order of tasks that minimizes the overall difficulty of the task order graph while satisfying two constraints.

## ACKNOWLEDGMENTS

This work is supported by the National Key R & D Program of China (2018YFC0809800).

## REFERENCES

- [1] C. Chai, L. Cao, G. Li, J. Li, Y. Luo, and S. Madden. Human-in-the-loop outlier detection. *In SIGMOD*, pages 19–33, 2020.
- [2] X. Chu, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, N. Tang, and Y. Ye. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. *In SIGMOD*, page 1247–1261, 2015.
- [3] S. Giannakopoulou, M. Karpathiotakis, B. C. Gaidioz, and A. Ailamaki. Cleanm: An optimizable query language for unified scale-out data cleaning. *In VLDB*, 2017.
- [4] C. Gokhale, S. Das, A. Doan, J. F. Naughton, N. Rampalli, J. Shavlik, and X. Zhu. Corleone: Hands-off crowdsourcing for entity matching. *In SIGMOD*, pages 601–612, 2014.
- [5] H. Park and J. Widom. Crowdfill: Collecting structured data from the crowd. *In SIGMOD*, pages 577–588, 2014.
- [6] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré. Holoclean: Holistic data repairs with probabilistic inference. *In VLDB*, pages 1190–1201, 2017.
- [7] Y. Xi, N. Wang, S. Hao, W. Yang, and L. Li. Pocketview: A concise and informative data summarizer. *In ICDE*, pages 1742–1745, 2020.