Demonstrating Matelda for Multi-Table Error Detection

Fatemeh Ahmadi
TU Berlin & BIFOLD
Berlin, Germany
f.ahmadi@tu-berlin.de

Julian Paulußen
TU Berlin & BIFOLD
Berlin, Germany
paulussen@tu-berlin.de

Ziawasch Abedjan
TU Berlin & BIFOLD
Berlin, Germany
abedjan@tu-berlin.de

ABSTRACT

Real-world datasets are often fragmented across multiple heterogeneous tables, managed by different teams or organizations. Ensuring data quality in such environments is challenging, as traditional error detection tools typically operate on isolated tables and overlook cross-table relationships. To address this gap, we investigate how cleaning multiple tables simultaneously, combined with structured user collaboration, can reduce annotation effort and enhance the effectiveness and efficiency of error detection.

We present Matelda, an interactive system for multi-table error detection that combines automated error detection with human-in-the-loop refinement. Matelda guides users through Inspection & Action, allowing them to explore system-generated insights, refine decisions, and annotate data with contextual support. It organizes tables using domain-based and quality-based folding and leverages semi-supervised learning to propagate labels across related tables efficiently. Our demonstration showcases Matelda's capabilities for collaborative error detection and resolution by leveraging shared knowledge, contextual similarity, and structured user interactions across multiple tables.

PVLDB Reference Format:
Fatemeh Ahmadi, Julian Paulußen, and Ziawasch Abedjan. Demonstrating Matelda for Multi-Table Error Detection. PVLDB, 18(12): 5379 - 5382, 2025.
doi:10.14778/3750601.3750676

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/D2IP-TUB/Matelda-Demo.

1 INTRODUCTION

With the increasing reliance on data-driven artificial intelligence systems, there is a growing need for data quality assurance. One of the research branches that deals with the improvement of data quality is data cleaning in terms of finding errors and correcting them [1].

As of today, there is a large body of research on practical tools for identifying and fixing errors [1]. In particular there are approaches for rule-based cleaning [4, 11], unsupervised approaches based on distributional patterns [10], and learning-based systems that require labeled training data [5, 7].

A limiting factor of many of the previous systems is that they are designed for a well-defined setup of cleaning a single relational table. Yet, in practice, multiple datasets or even entire databases are subject to data quality issues. In such multi-table scenarios, users must perform the typical cleaning tasks, such as writing rules or providing annotations for each table individually. This results in unnecessary repetition of effort, as users might need to perform similar tasks across tables, while the insights they generate remain isolated and are not effectively transferred to related tables [2].

Consider a scenario where a data steward is preparing census housing data for quality assessment. The datasets include hundreds of different tables that are separated at district-level and contain statistics from the Census, such as accommodation type and tenure. Although these tables describe related aspects of the same districts, they may differ in structure, terminology, and coverage. Some represent districts by name, others by code. Some treat missing values differently or categorize household types inconsistently. When the stewards attempts to curate the datasets, they have to annotate or define quality rules for each table, even when similar patterns or semantics might appear across datasets. This task can become even more cumbersome, as the number of tables grows, for example, to include historical census data, releases from other cities or countries, or data maintained by different agencies. Annotation without sufficient context is particularly challenging, especially when supporting evidence or signals reside in other tables. At the same time, many quality issues might be common across tables and annotations could be reused. Without the appropriate system support, users must manually rediscover these opportunities from the fragmented knowledge or will be subject to redundant efforts.

In this demonstration, we present Matelda, a framework for multi-table error detection that benefits from knowledge sharing across related datasets. Matelda combines automated detection with interactive user guidance, enabling users to inspect data, validate system suggestions, and annotate values with contextual support. The system helps users allocate labeling effort more efficiently and leverage inter-table relationships to support annotation tasks. Matelda introduces a workflow that organizes tables based on their semantic and quality-based similarities. A user-in-the-loop interaction model ensures that human expertise is effectively integrated into the error detection process. Through *Inspection & Action* layers, users explore raw data and system-generated signals, provide targeted annotations, and dynamically adjust their labeling and grouping strategy to balance effort and effectiveness.

During the demonstration, users can explore both synthetic and real-world datasets containing different types of errors. This allows them to experience the difference in annotation effort and system performance across various datasets.

As part of the demonstration, attendees will be invited to actively engage with Matelda through a responsive, web-based interface, accessible both via demo stations and on personal devices, for example by scanning a QR code with their phone. Users step into the

Proceedings of the VLDB Endowment, Vol. 18, No. 12 ISSN 2150-8097.
doi:10.14778/3750601.3750676

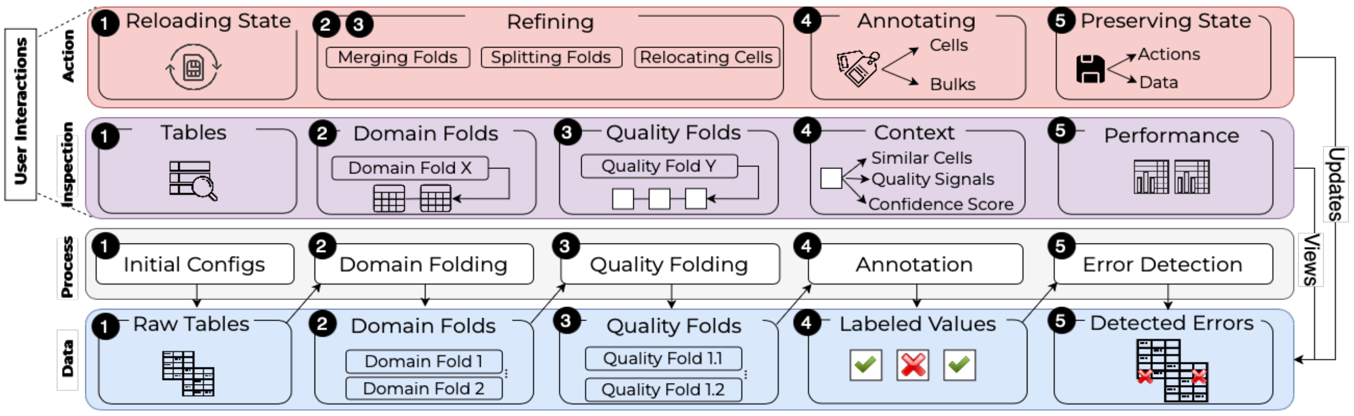


Figure 1: Overview of the Matelda pipeline and user interaction components.

role of a data steward and interactively label values across multiple tables within the aforementioned datasets. Matelda presents system-generated suggestions, which users can accept, reject, or refine, with each decision triggering updates across semantically or structurally related data points. This hands-on experience allows users to observe how annotations propagate across datasets, how context from one table supports labeling in another, and how their feedback, such as refining table groupings or annotation decisions, can influence the overall detection results. To emphasize Matelda's advantages, the demo also includes a baseline comparison: users can try a traditional one-table-at-a-time error detection workflow and experience the difference in annotation efficiency and multi-table context awareness.

2 SYSTEM OVERVIEW

In this demonstration, we highlight the main features of our multi-table error detection system **Matelda**. Matelda combines automated techniques with interactive user guidance. It leverages *domain-based* and *quality-based folding* to organize table fragments and incorporates *semi-supervised learning* to propagate user annotations and improve detection quality across related tables.

To support user-driven exploration and refinement, Matelda introduces structured interaction components based on two complementary modes: *Inspection* and *Action*. These modes guide how users engage with the system, explore system outputs, and iteratively refine results.

Figure 1 presents an overview of Matelda's architecture, organized into four aligned conceptual layers: ***Action Layer:*** An interface for user-driven interventions to guide the system. ***Inspection Layer:*** System-generated views on internal components to support user interpretation. ***Process Layer:*** The five key pipeline stages. ***Data Layer:*** Flow of data from raw tables to detected errors.

These layers facilitate to express Matelda's engaging approach for semi-supervised cleaning. In particular, we can highlight the relationship between system produced inspection elements and user-driven actions. We now describe this interaction and the system pipeline in more detail.

Matelda's pipeline consists of five main stages, each tied to the possible Inspection & Action interfaces:

Step 1: Dataset Initialization and Budget Allocation. Users begin by selecting a dataset, specifying a labeling budget, and selecting the base error detectors. They may also resume from a previously saved session. The prototype is able to save and reload sessions without rerunning the intermediate steps.
Step 2: Domain-Based Folding. Matelda groups tables by semantic similarity to organize tables into similar domain folds.
Inspection: Users can explore domain folds, table relationships, and drill-downs into specific table content.
Action: Users can intervene and refine folds by merging or splitting them and relocating tables as deemed appropriate.
Step 3: Quality-Based Folding. Within each domain, Matelda clusters cell values based on their quality characteristics that are delivered via base-detectors. These detectors are based on heuristics, such as outlier detection, inter-column relationships, and general purpose pattern violations. For more details we refer to the underlying full paper [2].
Inspection: Users can again investigate the fine-grained folds to validate their cohesiveness.
Action: Users can adjust folds to balance labeling effort and detection granularity.
Step 4: Annotation. Matelda selects representative cell values from each fold for labeling.
Inspection: Users can view detector outputs, metadata, and confidence scores to judge the proposed cells.
Action: Users can annotate values or apply bulk labels. Further the users can turn-off individual detectors that serve as features at will.
Step 5: Error Detection. Matelda detects and highlights erroneous cells based on user feedback.
Inspection: Users examine predicted errors and their confidence.
Action: Users can refine annotations and preserve system state.

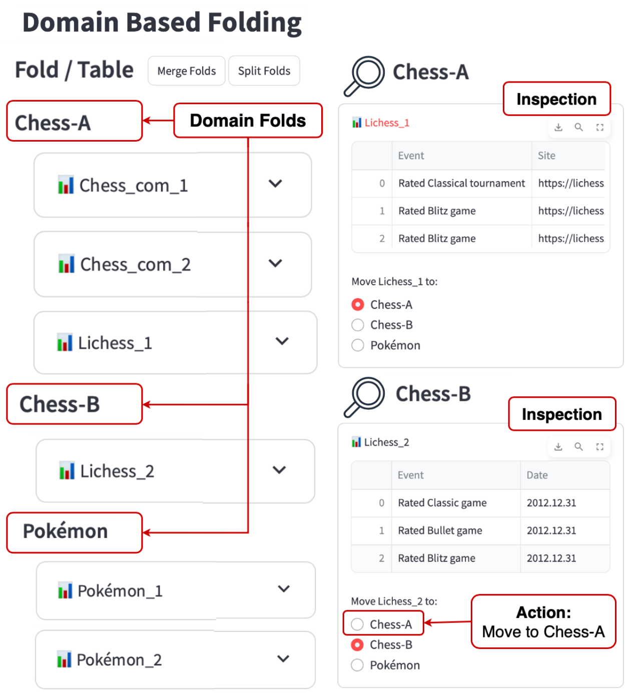
3 DEMONSTRATION

During the demonstration, we guide users through the individual steps of our pipeline as presented in the previous section. We built a user interface based on the Streamlit library [1] that enables both web-based interaction on desktops as well as phones [2].

[1]https://streamlit.io
[2]Matelda is the first cleaning tool that lets you swipe tables on your phone.

Table 1: Dataset characteristics. The error types are missing value (MV), typo (T), formatting issue (FI), violated attribute dependency (VAD) and numeric outliers (NO).

Name	Number of Tables	Size (#Cells)	Error Rate (cells)	Error Types
Quintet	5	199772	9%	MV, T, FI, VAD
DGov-NO	96	874570	2%	NO
DGov-Typo	96	874570	9%	FI & T
DGov-RV	96	874570	8%	VAD
WDC	100	64286	unknown	unknown



Lichess_1

	Event	Site
0	Rated Classical tournament	https://lichess
1	Rated Blitz game	https://lichess
2	Rated Blitz game	https://lichess

Move Lichess_1 to:
[✓ Chess-A]
Chess-B
Pokémon

Lichess_2

	Event	Date
0	Rated Classic game	2012.12.31
1	Rated Bullet game	2012.12.31
2	Rated Blitz game	2012.12.31

Move Lichess_2 to:
Chess-A
[✓ Chess-B]
Pokémon

Figure 2: Step 2 - Domain-Based Folding

To illustrate the user interface and the end-to-end workflow, we will walk through an example lake constructed from multiple tables sourced from Kaggle [3] and spanning two domains: Chess and Pokémon. The kaggle datasets come from multiple platforms and sources for each domain, for instance, the chess datasets come from Lichess and Chess.com. Further, users will be able to interact with a variety of datasets, including synthesized examples that exhibit different types of data quality issues, such as semantic and syntactic inconsistencies, as well as real-world data. Table 1 shows the characteristics of these datasets. With the diverse set of datasets, we showcase how the advantages of a multi-table approach varies depending on the nature of the underlying errors.

3.1 Demo Setup

We guide users through Matelda's key interaction steps.
Step 1: Initializations. The user selects the dataset or a previously preformed pipeline. They also specify a labeling budget which can be refined later. Also, base error detectors can be selected.

[3]https://www.kaggle.com/datasets

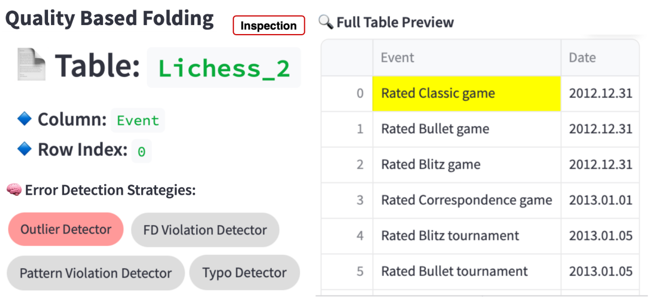
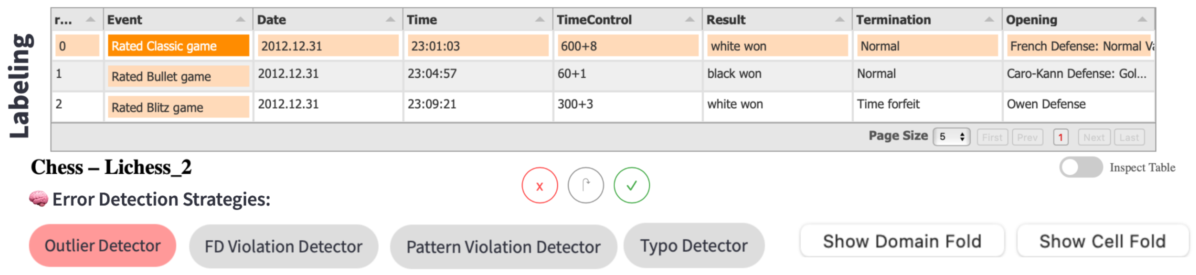


Table: Lichess_2
Column: Event
Row Index: 0
Error Detection Strategies:

	Event	Date
0	Rated Classic game	2012.12.31
1	Rated Bullet game	2012.12.31
2	Rated Blitz game	2012.12.31
3	Rated Correspondence game	2013.01.01
4	Rated Blitz tournament	2013.01.05
5	Rated Bullet tournament	2013.01.05

Figure 3: Step 3 - Quality-Based Folding

Step 2: Domain-Based Folding. Matelda automatically groups tables based on their semantic content. Figure 2 shows the user interface for this step. In our toy example, as Figure 2 shows, the system produces three folds: two corresponding to Chess-related tables and one containing the Pokémon-related tables. If needed, the user can refine the groupings by merging or splitting folds, or by reassigning tables across folds to improve semantic consistency. For example, as shown in Figure 2, the user moves the table Lichess_2 into the Chess-A fold, resulting in two semantically coherent folds.
Step 3: Quality-Based Folding. Within each domain fold, Matelda clusters cell values based on quality-related signals, such as formatting inconsistencies and violations of functional dependencies. The user explores these quality-based folds and inspects representative values as well as the context. Figure 3 shows the information that the user can see for each cell in each cell fold. They can also refine the folds and turn individual detectors on or off. For example, consider two Lichess tables within the Chess domain. The system groups cells from the Event attribute on both tables marked by the same detectors. Suppose we observe two groups: Group 1 includes "Rated Blitz game" and "Rated Bullet tournament"; Group 2 includes "Rated Classic game" and "Rated Classic tournament". No detectors flag the values in Group 1, whereas those in Group 2 are marked by the outlier detection tool.
Step 4: Cross-Table Annotation. Matelda selects representative cell values for labeling based on quality-based folds. The user explores values alongside metadata such as column descriptions, value frequency, and quality signals. They can also examine neighboring values within a fold and compare similar cell values across tables to identify shared patterns. For example, in Figure 4, the selected cell "Rated Classic game" is marked by outlier detectors. However, without looking into the other table from Lichess, the user would not be able to easily detect that this cell is erroneous and the value should be "Rated Classic**al** game". Based on this context, the user labels representative values or applies bulk annotations to similar cells within and across tables. In this case, the label will also be propagated to "Rated Classic tournament" via the system's propagation mechanisms, as both are in the same quality fold.
Step 5: Error Detection. The system trains classifiers based on the user's annotations and predicts whether cell values are erroneous. The user reviews the predicted errors along with system confidence scores and evaluation scores. The user can compare the results with similar pipelines on the same dataset and saves their progress for continued analysis.

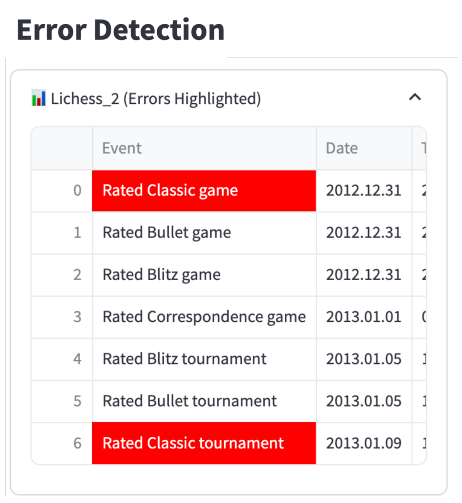


r...	Event	Date	Time	TimeControl	Result	Termination	Opening
0	Rated Classic game	2012.12.31	23:01:03	600+8	white won	Normal	French Defense: Normal Va
1	Rated Bullet game	2012.12.31	23:04:57	60+1	black won	Normal	Caro-Kann Defense: Gol...
2	Rated Blitz game	2012.12.31	23:09:21	300+3	white won	Time forfeit	Owen Defense

Chess – Lichess_2

🧠 Error Detection Strategies:

Figure 4: Step 4 - Context-Aware Annotation


Error Detection

📊 Lichess_2 (Errors Highlighted)

	Event	Date
0	Rated Classic game	2012.12.31
1	Rated Bullet game	2012.12.31
2	Rated Blitz game	2012.12.31
3	Rated Correspondence game	2013.01.01
4	Rated Blitz tournament	2013.01.05
5	Rated Bullet tournament	2013.01.05
6	Rated Classic tournament	2013.01.09

Figure 5: Step 5 - Error Detection Results on a Mobile Screen

Outcome: This demonstration scenario highlights how Matelda supports users in navigating the complexity of annotation in multi-table settings. By combining semantic domain grouping, quality-based folding, and interactive refinements, users can efficiently annotate related tables while leveraging cross-table patterns. Unlike prior systems, Matelda offers structured interactions that reduce redundancy and enhance annotation confidence.

3.2 Accessibility and Engagement

Matelda is implemented as a responsive web application, supporting both desktop and mobile devices. Figure 5 shows that the interface adapts to different screen sizes, ensuring that users can explore folds, inspect metadata, and annotate values across platforms. This flexibility enables both casual inspection on-the-go and in-depth analysis in desktop environments.

4 RELATED WORK

There have been several prior works on data cleaning that have been demonstrated [3, 4, 7, 9, 11]. The primary objective of these demonstrations was to showcase how a system can interactively clean a single table with user assistance. While our work is inspired by the example-driven approaches [6, 8], the focus is on multi-table data cleaning, which introduces different challenges and requirements. In single-table setting, users need to be familiar with the context of the provided table to be able to provide rules [4, 9], validate them [3], or annotate examples [7]. Our demonstration is innovative in the sense that it addresses error detection in a multi-table scenario. it is based on our recently published paper [2] and includes a wide range of interaction options tailored to the new setup and algorithm as described in the previous section. It enables navigation across tables and inspection and manipulation of annotation sharing for semi-supervised cleaning.

ACKNOWLEDGMENTS

This project has been supported by the German Research Foundation (DFG) under grant agreement 387872445.

REFERENCES

[1] Ziawasch Abedjan, Xu Chu, Dong Deng, Raul Castro Fernandez, Ihab F. Ilyas, Mourad Ouzzani, Paolo Papotti, Michael Stonebraker, and Nan Tang. 2016. Detecting Data Errors: Where are we and what needs to be done? *Proceedings of the VLDB Endowment (PVLDB)* (2016).
[2] Fatemeh Ahmadi, Marc Speckmann, Malte F. Kuhlmann, and Ziawasch Abedjan. 2025. MaTElDa: Multi-Table Error Detection. In *Proceedings of the International Conference on Extending Database Technology (EDBT).*
[3] Xu Chu, Mourad Ouzzani, John Morcos, Ihab F. Ilyas, Paolo Papotti, Nan Tang, and Yin Ye. 2015. KATARA: Reliable Data Cleaning with Knowledge Bases and Crowdsourcing. *Proceedings of the VLDB Endowment (PVLDB)* (2015).
[4] Amr Ebaid, Ahmed K. Elmagarmid, Ihab F. Ilyas, Mourad Ouzzani, Jorge-Arnulfo Quiané-Ruiz, Nan Tang, and Si Yin. 2013. NADEEF: A Generalized Data Cleaning System. *Proceedings of the VLDB Endowment (PVLDB)* (2013).
[5] Alireza Heidari, Joshua McGrath, Ihab F. Ilyas, and Theodoros Rekatsinas. 2019. HoloDetect: Few-Shot Learning for Error Detection. In *Proceedings of the International Conference on Management of Data (SIGMOD).*
[6] Mohammad Mahdavi and Ziawasch Abedjan. 2020. Baran: Effective Error Correction via a Unified Context Representation and Transfer Learning. *Proceedings of the VLDB Endowment (PVLDB)* (2020).
[7] Mohammad Mahdavi and Ziawasch Abedjan. 2021. Semi-Supervised Data Cleaning with Raha and Baran. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR).*
[8] 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 *Proceedings of the International Conference on Management of Data (SIGMOD).*
[9] Sebastian Schelter, Felix Bießmann, Dustin Lange, Tammo Rukat, Philipp Schmidt, Stephan Seufert, Pierre Brunelle, and Andrey Taptunov. 2019. Unit Testing Data with Deequ. In *Proceedings of the International Conference on Management of Data (SIGMOD).*
[10] Pei Wang and Yeye He. 2019. Uni-Detect: A Unified Approach to Automated Error Detection in Tables. In *Proceedings of the International Conference on Management of Data (SIGMOD).*
[11] Mohamed Yakout, Ahmed K. Elmagarmid, Jennifer Neville, and Mourad Ouzzani. 2010. GDR: a system for guided data repair. In *Proceedings of the International Conference on Management of Data (SIGMOD).*