Published: 19 March 2024
Online AM: 15 February 2024
Accepted: 25 January 2024
Revised: 23 January 2024
Received: 14 March 2023

Citation in BibTeX format

RESEARCH-ARTICLE

Cleenex: Support for User Involvement during an Iterative Data Cleaning Process

JOÃO L PEREIRA, University of Évora, Evora, Evora, Portugal

MANUEL J. FONSECA, University of Lisbon, Lisbon, Lisbon, Portugal

ANTÓNIA LOPES, University of Lisbon, Lisbon, Lisbon, Portugal

HELENA GALHARDAS, Higher Technical Institute, Lisbon, Lisbon, Portugal

Cleenex: Support for User Involvement during an Iterative Data Cleaning Process

JOÃO L. M. PEREIRA, VISTA Lab, Algoritmi Center, University of Évora, Portugal
MANUEL J. FONSECA, LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal
ANTÓNIA LOPES, LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal
HELENA GALHARDAS, INESC-ID and Instituto Superior Técnico, Universidade de Lisboa, Portugal

The existence of large amounts of data increases the probability of occurring data quality problems. A data cleaning process that corrects these problems is usually an iterative process, because it may need to be re-executed and refined to produce high-quality data. Moreover, due to the specificity of some data quality problems and the limitation of data cleaning programs to cover all problems, often a user has to be involved during the program executions by manually repairing data. However, there is no data cleaning framework that appropriately supports this involvement in such an iterative process, a form of human-in-the-loop, to clean structured data. Moreover, data preparation tools that somehow involve the user in data cleaning processes have not been evaluated with real users to assess their effort.

Therefore, we propose Cleenex, a data cleaning framework with support for user involvement during an iterative data cleaning process, and conduct two data cleaning experimental evaluations: an assessment of the Cleenex components that support the user when manually repairing data with a simulated user; and a comparison, in terms of user involvement, of data preparation tools with real users.

Results show that Cleenex components reduce the user effort when manually cleaning data during a data cleaning process, for example, the number of tuples visualized is reduced in 99%. Moreover, when performing data cleaning tasks with Cleenex, real users need less time/effort (e.g., half the clicks) and, based on questionnaires, prefer it to the other tools used for comparison, OpenRefine and Pentaho Data Integration.

CCS Concepts: • **Information systems** → **Data cleaning**; • **Human-centered computing** → *User studies*; *Usability testing*; *Interaction paradigms;*

Additional Key Words and Phrases: Data quality, data curation, user involvement, human-in-the-loop

This work was performed while João L. M. Pereira was a PhD student at INESC-ID and IST, Universidade de Lisboa.
This work was supported by *Fundação para a Ciência e a Tecnologia* through the PhD Scholarship, ref. SFRH/BD/135719/2018; through INESC-ID research organization, ref. UIDB/50021/2020 (DOI:10.54499/UIDB/50021/2020); and through the LASIGE research unit, ref. UIDB/00408/2020 (DOI:10.54499/UIDB/00408/2020) and ref. UIDP/00408/2020 (DOI:10.54499/UIDP/00408/2020).
Authors' addresses: J. L. M. Pereira, VISTA Lab, Algoritmi Center, University of Évora, Rua Romão Ramalho nº59, Évora, 7000-671, Portugal; e-mail: joao.pedro.pereira@uevora.pt; M. J. Fonseca and A. Lopes, LASIGE, Faculdade de Ciências, Universidade de Lisboa, Faculdade de Ciências da Universidade de Lisboa, Campo Grande, 1749-016, Lisbon, Portugal; e-mails: mjfonseca@ciencias.ulisboa.pt, malopes@ciencias.ulisboa.pt; H. Galhardas, INESC-ID and Instituto Superior Técnico, Universidade de Lisboa, IST Tagus Park, Avenida Professor Cavaco Silva, Porto Salvo, 2780-990, Portugal; e-mail: helena.galhardas@tecnico.ulisboa.pt.

ACM 1936-1955/2024/03-ART6
https://doi.org/10.1145/3648476

ACM Reference Format:
João L. M. Pereira, Manuel J. Fonseca, Antónia Lopes, and Helena Galhardas. 2024. Cleenex: Support for User Involvement during an Iterative Data Cleaning Process. *ACM J. Data Inform. Quality* 16, 1, Article 6 (March 2024), 26 pages. https://doi.org/10.1145/3648476

1 INTRODUCTION

Nowadays, data is an essential asset of any business or public organization to support decision through data analysis or automatic recommendations. Moreover, data is the basis of the main products of IT companies such as Google, Facebook, Spotify, and IMDb. To guarantee successful decisions, data quality is essential, otherwise, we end up with the known situation: "garbage in garbage out." In this context, garbage consists of data quality problems that result from a variety of situations, from human error and sensor malfunctions to data integration scenarios.

Poor data quality greatly impacts data-based decisions. The world's top companies have more than 25% of its critical data dirty [50]. The existence of dirty data may have disastrous consequences. For instance, wrong Electronic Health Records can lead to bad assumptions about a certain treatment. McKinsey estimates savings of 300 billion dollars every year in the United States of America's health care [29] by properly analyzing and correcting health data. Globally, Gartner measures the average impact of data quality in organizations as $9.7 million per year [49].

Data quality problems that exist in structured data (e.g., relational databases) may be of different types [32]. Single value data quality problems enclose, for instance, missing values, incorrect values (e.g., syntax violations, misspelled errors, domain constraint violations), and ambiguous values (e.g., acronyms with no expansion). When considering a set of records and/or attributes, we can find violations of domain constraints (e.g., a publication published in year 3000), inconsistencies (e.g., two publications with the same journal name and volume number that are not published in the same year), and approximate duplicates (e.g., two records referring to the same person).

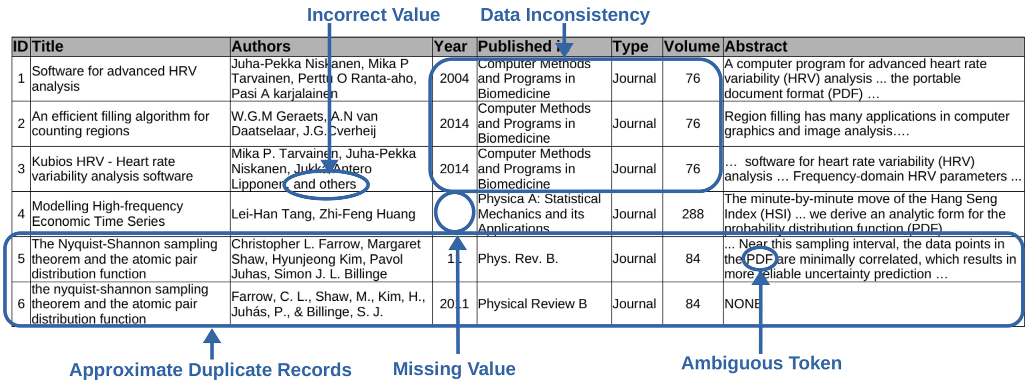
Example 1.1. Consider Figure 1 that presents a table containing structured data about publications. Some quality problems can be observed: (i) the incorrect value "and others" in the list of authors for publication record 3; (ii) the data inconsistency that involves records 1, 2, and 3 given that they do not satisfy the condition that journal publications published in the same journal and volume must have the same publication year; (iii) a missing value in the year field of publication 4; (iv) the occurrence of the acronym PDF in the abstract field of publication 5, with no expansion and thus consisting on an ambiguous token; and (v) approximate duplicate records 5 and 6 referring to the same publication.

1.1 Cleaning Structured Data

Data cleaning is today acclaimed as a central feature of data analysis and management tools. Data cleaning aims at converting source data into target data without errors, missing values, ambiguities, duplicates, and inconsistencies [39]. It is thus of paramount importance to execute a data cleaning process to effectively eliminate data quality problems [46].

A data cleaning program to clean structured data is typically modeled by a program designer as: (a) a graph of data transformations [16] or (b) a set of data quality rules [15].

In the approach (a), a data cleaning program is modeled as a graph of data cleaning transformations whose execution produces cleaned data. By providing distinct and configurable data operators that can be composed to implement those data transformations, data cleaning programs based on graphs of data transformations can clean data inconsistencies, domain/syntax constraints, approximate duplicate records/attributes, missing values, and incorrect data if the correct values are available in a reliable data source, which is integrated in the data cleaning process.


Incorrect Value
Data Inconsistency
Approximate Duplicate Records
Missing Value
Ambiguous Token

ID	Title	Authors	Year	Published in	Type	Volume	Abstract
1	Software for advanced HRV analysis	Juha-Pekka Niskanen, Mika P Tarvainen, Perttu O Ranta-aho, Pasi A karjalainen	2004	Computer Methods and Programs in Biomedicine	Journal	76	A computer program for advanced heart rate variability (HRV) analysis ... the portable document format (PDF) …
2	An efficient filling algorithm for counting regions	W.G.M Geraets, A.N van Daatselaar, J.G.Cverheij	2014	Computer Methods and Programs in Biomedicine	Journal	76	Region filling has many applications in computer graphics and image analysis.…
3	Kubios HRV - Heart rate variability analysis software	Mika P. Tarvainen, Juha-Pekka Niskanen, Jukka Antero Lipponen, and others	2014	Computer Methods and Programs in Biomedicine	Journal	76	… software for heart rate variability (HRV) analysis … Frequency-domain HRV parameters ...
4	Modelling High-frequency Economic Time Series	Lei-Han Tang, Zhi-Feng Huang		Physica A: Statistical Mechanics and its Applications	Journal	288	The minute-by-minute move of the Hang Seng Index (HSI) ... we derive an analytic form for the probability distribution function (PDF)
5	The Nyquist-Shannon sampling theorem and the atomic pair distribution function	Christopher L. Farrow, Margaret Shaw, Hyunjeong Kim, Pavol Juhas, Simon J. L. Billinge	11	Phys. Rev. B.	Journal	84	... Near this sampling interval, the data points in the PDF are minimally correlated, which results in more reliable uncertainty prediction …
6	the nyquist-shannon sampling theorem and the atomic pair distribution function	Farrow, C. L., Shaw, M., Kim, H., Juhás, P., & Billinge, S. J.	2011	Physical Review B	Journal	84	NONE

Fig. 1. Example of data quality problems in a publications table.

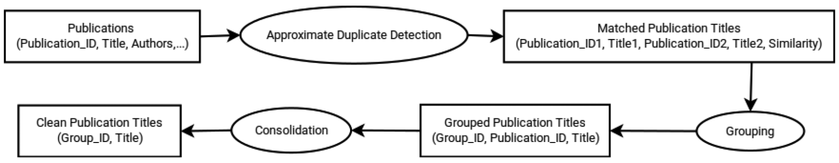

Publications
(Publication_ID, Title, Authors,...)
Approximate Duplicate Detection
Matched Publication Titles
(Publication_ID1, Title1, Publication_ID2, Title2, Similarity)
Grouping
Grouped Publication Titles
(Group_ID, Publication_ID, Title)
Consolidation
Clean Publication Titles
(Group_ID, Title)

Fig. 2. Part of a graph from a data transformation cleaning program.

Example 1.2. In Figure 2, we present part of a short data cleaning program based on data transformations. In this graph, ellipses represent high-level data transformations, rectangles represent data tables, and arrows represent data flows. This data cleaning program aims at identifying and consolidating approximate duplicate scientific publications similar to records 5 and 6 identified in Figure 1. First, the data cleaning program executes an *Approximate Duplicate Detection* transformation that computes the similarity among all possible publication title pairs and filters out the non-similar pairs of titles. Second, it applies a *Grouping* transformation to similar pairs of titles, which assigns a *Group_ID* to each publication title using a transitive closure algorithm, thus guaranteeing that similar publications share the same *Group_ID*. Third, it applies a *Consolidation* transformation to find the best representative publication title for each group of publications that share the same *Group_ID*.

In the second type of data cleaning programs (b), *data quality rules* express conditions that data must satisfy to be considered of good quality. If data does not satisfy a rule, then a violation occurs and a data repair that corrects the data must be found. Data repairs are typically selected from the set of possible data repairs using heuristics. In the literature, several formalisms were proposed to express data quality rules. For example, the authors of Reference [6] proposed **Conditional Functional Dependencies (CFDs)**, an extension to Functional Dependencies [13].

By exploring different types of formalisms to specify data quality rules, these rule-based data cleaning programs can identify and possibly lead to the resolution of data quality problems.

Example 1.3. To detect the data inconsistency that involves records 1, 2, and 3 in Figure 1, we can use a CFD that represents the rule: All publications whose *Type* value is "Journal" and have the same *Published In* and *Volume* values, must have the same year value. Translating to CFD notation, this would correspond to: `Type[Journal], Published In, Volume -> Year`. In this situation, data cleaning consists in generating, selecting, and applying a data repair that modifies the publications records to satisfy the CFD. Possible data repairs are: (i) change the Year of tuple 1 to 2014 or (ii) change the Volume value of record 1 to a different value (e.g., 50).

It is worth noting that commercial data cleaning tools typically support data cleaning graphs [3], which are useful for users to control the data flow [24]. Nevertheless, the majority of recent research contributions rely on complex data quality rules and on the expensive generation of data repairs [24].

1.2 User Feedback in a Data Cleaning Process

Often, the execution of a data cleaning program is insufficient to clean all the data quality problems present in real databases. This happens because: (i) the data quality rules or transformations available cannot cover all the cleaning criteria required to generate clean data, leaving some dirtiness; or (ii) in case of machine learning-based data cleaning, the available clean data may not be sufficient to infer the correct values (e.g., infer the name of a publication). For instance, missing and incorrect values (e.g., missing year) cannot be cleaned by data transformations nor generated data repairs without using an external data source that contains the correct values. Another example is when detecting approximate duplicates. In practice, there is no approximate duplicate criteria (i.e., similarity function and threshold) that perfectly identifies records that refer to the same entity in the real world. The incorrect identification of approximate duplicate records results in false positives or false negatives thus leaving some data dirtiness.

After a data cleaning process, users with domain knowledge typically have to manually repair some data tuples to guarantee the highest quality levels of the resulting data. In these cases, enabling a user to manually clean a subset of data is crucial.

1.3 Support for the User Involvement during the Execution of a Data Cleaning Process

In this work, we focus on a particular aspect of human-in-the-loop for data cleaning: user involvement during the execution of a data cleaning process that is modeled as a data cleaning graph. User involvement is required during the execution of a data cleaning process because: (i) data transformations used in a data cleaning process need to be tuned, or (ii) data records that are generated by the transformations that compose a data cleaning process may need to be manually repaired through user feedback during the execution of an automatic data cleaning process.

Similarly to software development, a data cleaning process needs to be iteratively refined and executed to obtain the highest quality of data. These refinements can be due to: (i) new data batches that are continuously provided as input and contain new data quality problems or (ii) modifications to data transformations that may introduce new data quality problems. Unlike software development, a data cleaning process often benefits from manually data repairs where the user feedback resolves data quality problems, e.g., using expert knowledge to impute missing values. Moreover, if these repairs are introduced in early phases of the data cleaning process, then they can avoid the propagation of data quality problems to further stages of the process.

In the literature, Rezig et al. [45] describe a design vision for an end-to-end data cleaning framework that includes several types of user involvement during a data cleaning process, such as refining a data cleaning program and user feedback to manually clean data. Other complementary tasks in data quality such as data profiling and exploration have benefited from user involvement to detect data quality problems and efficiently explore data [37, 51].

Despite its importance in a data cleaning process, user involvement in commercially adopted data preparation tools to perform data cleaning is limited. **ETL (Extract, Transform, and Load)** tools like **Pentaho Data Integration (PDI)**[1] only support the user to tune and combine data transformations. Data wrangling tools like OpenRefine[2] support manual data repairs and automatic data transformation. However, the single possibility to modify applied manual data repairs and data transformation in a data cleaning program is to remove them, but that is also limited by order of newest to oldest in a similar manner to the Undo function present in word-processing software. So far, commercial ETL and data wrangling tools do not support the incorporation of user feedback to manually repair data during the iterative refinement and execution of a data cleaning process.

To summarize, currently, there is no data cleaning framework that supports, in a principled way, the user involvement during an iterative execution of such a process. Furthermore, no evaluation of ETL and data wrangling tools with real users to measure the user effort when designing data cleaning programs and manually correcting structured data has been performed so far.

1.4 Main Contributions

This article describes the following contributions that address the above mentioned problems regarding the support for the user involvement and user effort evaluation:

- **Cleenex**, a data cleaning framework that is based on the Ajax data cleaning framework [16] that provides customizable data transformations to clean relational databases. It extends Ajax with a *manual data repair management component* whose main principles were proposed in Reference [17]. This component allows the designer to specify where and how data can be manually repaired during the execution of a data cleaning process. Additionally, Cleenex offers a debugging mechanism for data cleaning processes that provides the provenance of the data. To support the user involvement in an iterative execution of a data cleaning process, we developed two additional components (*manual data repair persistence* and *manual data repair recovery*) that prevent the integral re-execution of data cleaning processes and the loss of previous manual data repairs. Moreover, we formulated and implemented an *algorithm for the detection and automatic recovery of manual data repairs*, that, given a new execution of a data cleaning process, identifies which manual data repairs can be automatically re-applied. Even when manual data repairs cannot be automatically re-applied, Cleenex attempts to recover those manual data repairs by requesting the user for additional feedback, which is less demanding than manually repairing the data from scratch.
- An **extensive user involvement evaluation of Cleenex** that includes two evaluation studies: (i) the *evaluation of the three individual Cleenex components that support the user involvement* (i.e., manual data repairs, persistence, and recovery); and (ii) the *evaluation of Cleenex against two other data preparation tools*, OpenRefine and PDI, in terms of their efficiency and effectiveness to support the user involvement.
  To evaluate the Cleenex components, we programmed a simulated and ideal user to execute a data cleaning process, to manually repair the data, and to provide feedback during such an iterative process. We measured the required amount of data that the simulated user has to visualize and the number of actions that the simulated user has to perform to clean the data.
  To evaluate Cleenex against OpenRefine and PDI, we performed a within-subjects experimental evaluation with 32 users that include: (i) two datasets with data quality problems that simulate real scenarios, (ii) two tasks in the context of data cleaning that need to be executed by users using each tool, and (iii) a satisfaction questionnaire that assessed the user perception about each tool. The execution of the data cleaning tasks was evaluated by its

[1] http://community.pentaho.com/projects/data-integration/
[2] https://openrefine.org/

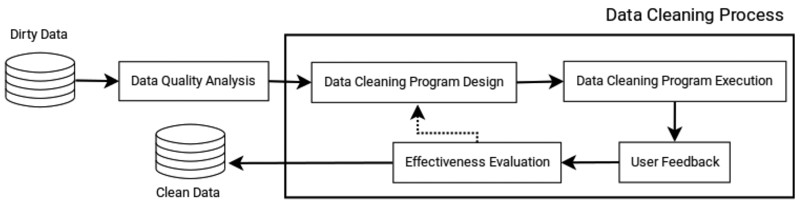


Fig. 3. Typical data cleaning process.

correct or incorrect completion, execution time, and number of user actions (e.g., clicks and keys pressed).

1.5 Document Outline

The rest of this document is structured as follows: Section 2 presents the fundamental concepts of data cleaning. Section 3 summarizes the most important related work about user involvement in cleaning data and data cleaning frameworks. In Section 4, we describe Cleenex and the components that support the user involvement in an iterative execution of a data cleaning process. In Section 5, we detail the experimental evaluation procedure and corresponding results to assess user involvement in such a process. In Section 6, we discuss the results and the limitations. Section 7 draws key conclusions and ideas for future work.

2 OVERVIEW OF A DATA CLEANING PROCESS

In Figure 3, we present a typical data cleaning process. Usually, this process is preceded by a *Data Quality Analysis* process that is intended to audit the *Dirty Data* through data quality rules and statistical techniques. Then, the user should have sufficient information to proceed with the *Data Cleaning Program Design* (with data transformations or data quality rules) that can clean the data quality problems previously identified. The next step is the *Data Cleaning Program Execution* followed by a step where the user can provide *User Feedback* by manually correcting instances of data quality problems not addressed by the automatic methods. In the *Effectiveness Evaluation* phase, the effectiveness of the data cleaning process is checked by measuring the data quality of a sample taken from the processed data. If the data is not sufficiently clean, then the same process is reapplied to the entire *Dirty Data* with a refined data cleaning program until the desired data quality is achieved. In the next sections, we present the two main types of data cleaning approaches. First, we describe data transformations in Section 2.1 and then data quality rules in Section 2.2.

2.1 Data Transformations

In this section, we describe a data cleaning process based on data transformations. A data cleaning program constituted of data transformations is usually designed based on a drag-and-drop graphical user interface, identical to most user interfaces supported by ETL software, where data transformations are represented by boxes and the data flows by links between those boxes.

A data cleaning program based on data transformations is usually modeled as a graph of data transformations. The effective combination of data transformations within a graph and the proper tuning of the cleaning criteria underlying each data transformation can transform dirty tables into clean tables.

Example 2.1. Consider publication references in a database table, obtained using an extraction software, we now want to obtain a list of approximate duplicate author name pairs. We start from an instance of this database table named *PublicationAuthors* as in Figure 4(a) that contains two attributes: (i) *Pub ID*, the publication identifier, and (ii) *Authors*, containing the last names of each

PublicationAuthors

Pub ID	Authors
1	Kanamori, Heiken
2	Kaanamori
3	Reeken, Kanamori
4	Knamoqri, Yothers

(a) Initial table.

Authors

Author ID	Name
1	Kanamori
2	Heiken
3	Kaanamori
4	Reeken
5	Kanamori
6	Knamoqri
7	Yothers

(b) List of author names.

SimilarAuthors

Author ID 1	Name 1	Author ID 2	Name 2	Distance
1	Kanamori	3	Kaanamori	1
1	Kanamori	5	Kanamori	0
1	Kanamori	6	Knamoqri	2
2	*Heiken*	*4*	*Reeken*	*2*
3	Kaanamori	5	Kanamori	1
5	Kanamori	6	Knamoqri	2

(c) Pairs of similar author names.

Fig. 4. Stage tables from transforming a table containing authors name by publication into a table containing similar author names.

publication author separated by commas. The final expected result is the *SimilarAuthors* table, shown in Figure 4(c), whose schema has 5 attributes: a pair of authors, each represented by an identifier *Author ID 1/2* and its last name *Name 1/2*; and a distance value between the two authors' last names within the same pair. To obtain such table, we have to execute an Approximate Duplicate Detection operation composed by the following two high-level data transformations:

Splitting: Splits each field value into a different row using a character or pattern as a separator. In the example, Splitting extracts the individual author names from the string field *PublicationAuthors.Authors* using a comma as a separator (Figure 4(a) to Figure 4(b)).

Approximate Duplicate Detection: Computes a distance value using a distance function (e.g., the edit-distance [27] function) for all possible pairs of values for the input field *Authors.Name* (in Figure 4(b)). Then, it filters the rows using a given threshold value, e.g., distance values below 3 (Figure 4(b) to Figure 4(c)).

Finally, to emphasize the need for the user to *manually correct instances* of data quality problems not addressed by the automatic method, note that, in Figure 4(c), the names *Heiken* and *Reeken* (tuple in Italic) do not refer to the same real author and so tuple needs to be removed.

2.2 Data Quality Rules

Data quality rules express conditions that data must satisfy to be considered of good quality. When a rule is violated, then a data quality problem occurs. In this section, we describe the type of rules available for data quality analysis and the data cleaning process based on such rules.

Conceptually, the first rules implemented in data management software (e.g., **RDBMS—Relational Database Management Systems)** were *integrity constraints*. Integrity constraints describe conditions among data values, for instance, a normal employee cannot earn more than the director of a company. ***Functional Dependencies*** **(FDs)** [13] are integrity rules that check if, for any set of tuples whose values of a set of user specified attributes X are equal (e.g., same Zipcode), the values of those tuples in another specified set of attributes Y (e.g., same City) are equal. An FD is represented as X `->` Y. As an example, consider a table Customers storing person records that contain location-related attributes such as City, Zipcode, and Country. We may state that if two or more records share the same zip code, then they must have the same city value. The corresponding FD is: `Zipcode -> City`. ***Conditional Functional Dependencies*** **(CFDs)** [6] are extensions to FDs that enable to use value conditions for attributes. Consider that the FD previously defined was only valid in the United States of America, then we can specify a CFD that expresses that condition as follows: `Country[United States of America], Zipcode -> City`. Other formalisms are: ***Inclusion Dependencies*** **(INDs)**, which express conditions that data records must satisfy across relations (e.g., a foreign key in a database); ***Conditional Inclusion Dependencies*** **(CIDs)** [7],

Table 1. Summary of Related Work on User Involvement in Data Cleaning and General Purpose Data Cleaning Tools

Related Work	Year	User Feedback to/for data repairs	Data Cleaning Approach
Ajax [16]	2001	No	Data transformations
FEBRL [9]	2005	No	Data transformations
GDR [55]	2011	Select data repairs	Automatic selection of data repairs for CFDs
LLUNATIC [18]	2013	Resolve conflicting data repairs	Automatic selection of data repairs for EGDs
NADEEF [11]	2013	No	Automatic selection of data repairs for CFDs, Mds, and DCs
CDC [52]	2014	Select repair types	Automatic selection of data and FDs repairs
FALCON [20]	2016	Provide data repairs and select update statements	Guide user to obtain data repairs
ActiveClean [25]	2016	Manual edit data values	Obtain edited data to improve ML models
Holoclean [43]	2017	No	Automatic generation of data repairs with probabilistic models, DCs, and knowledge bases
DANCE [2]	2018	Provide data repairs	Guide user to obtain data repairs to TGDs and EGDs
ICARUS [40]	2018	Provide data repairs and select update statements	Guide user to obtain data repairs
PIClean [56]	2019	Manual edit data and select data repairs	Automatic generation of data repairs with probabilistic models
UDATA [38]	2019	Input/output data examples and verify transformations	Data transformations
CoClean [30]	2020	Manual edit data values	Support multiple users to edit data simultaneously
Baran [28]	2020	Manual edit data values	ML models to edit data cells
Horizon [44]	2021	No	Automatic selection of data repairs for FDs
Garf [34]	2022	No	Automatic generation of data repairs with probablistic models
Räth et al. [42]	2023	Select suggested data transformations	Data transformations for streaming data
Precisely Data Integrity Suite	N/A	No	Data transformations
PDI	N/A	No	Data transformations
OpenRefine	N/A	Manual edit data values and delete tuples	Data transformations

which enable to express conditions over INDs (like the CFDs did with FDs); and ***Matching Dependencies*** **(MDs)** [14], which are dependencies designed for approximate duplicate detection. Finally, ***equality generating dependencies*** **(egds)** [4] consists on a formalism that standardize dependencies by proposing a syntax based on logic that encloses most of the data dependencies from the literature (e.g., CFDs, CIDs).

As stated above, data must satisfy the conditions described by a set of data quality rules, otherwise a violation occurs, meaning that data quality problems are found. To resolve a violation, there are different alternatives to modify the data called *data repairs*. For example, consider the records 1, 2, and 3 in Figure 1 and the CFD `Type[Journal], Published In, Volume -> Year`, an obvious data repair is to change the Year of tuple 1 to 2014. Another valid data repair would be to change the volume value of record 1 to a different value such as 50. Choosing which data repair to apply is usually performed through heuristics [8, 10, 11, 18].

3 RELATED WORK

In this section, we describe the related work in data cleaning. In Table 1, we first summarize research works that involve the user in a data cleaning process in such a way that she can provide feedback to repair the data, and then we overview both commercial and research general purpose data cleaning tools. We describe each work regarding: *(i)* its support for user feedback to repair data and *(ii)* the type of data cleaning approach. Additionally, we discuss the **Large Language Models (LLMs)** opportunities in the data cleaning field and other research directions.

User involvement. In LLUNATIC [18], data quality constraints are supported by a language based on equality generating dependencies (egds) [4] whose semantic incorporates most of the data quality rules from the literature. The user aims at resolving conflicts unresolved by automatic generated repairs for violations of data quality constraints. So, the user provides a value for cells marked with lluns. UDATA [38] learns data transformations through an input/output example, i.e., a set of dirty input tuples and desired (cleaned) output tuples provided by the user. Additionally, it asks the user to verify the generated data transformations. Although manual cleaning is performed by a user, it is only over a subset of data to serve as an example to generate data transformations.

In another line of work, **Guided Data Repairs (GDR)** [55] take advantage of the user to replace heuristics for selecting data repairs to solve **Conditional Functional Dependencies (CFDs)** violations while in **Continuous Data Cleaning (CDC)** [52] the user has to train a model that to solve a given violation decides whether to repair the data or to specialize the **Functional Dependencies (FDs)**. PiClean [56] and Baran [28] train **Machine Learning (ML)** and/or probabilistic models with user feedback from manual editing data. In PiClean, users also select the right data repairs, while Baran focuses only on modifications to single cells. ActiveClean [25] also learns from user feedback when editing data but aims at improving the data specifically for ML models by selecting the data to manually repair that maximizes ML models' performance.

DANCE [2], FALCON [20], ICARUS [40], and CoClean [30] aim at achieving high data quality by refusing the automatic selection of data repairs. Thus, they delegate data repairing exclusively to the user. DANCE tries to clean data by relying only on the user to produce data repairs. It minimizes the user effort by guiding her through the tuples that are the most likely to need data repairs. FALCON and ICARUS allow the user to first edit a cell and then generate general update statements based on the user modification. FALCON supports the edition of missing or incorrect values and focuses on minimizing the user effort in generating update statements, whereas ICARUS supports only the edition of missing values but obtains better performance than FALCON. Moreover, ICARUS focuses on selecting the initial data to present to the user. CoClean gives support to multiple users to edit a table at the same time.

Recently, Räth et al. [42] proposed to address iterative data cleaning when new data (i.e., stream data) are given as input, similar to our motivation. However, the authors' focus is on suggesting data transformations to this new data and, similarly to Potter's Wheel framework [41], considers user involvement to select such data transformations. Our work focuses on the user feedback when manually editing data.

So far, in the context of the user involvement in a data cleaning process, the user has been proposed to help resolve data quality constraint violations, to generate update statements (which can also be constraints) based on edited cells or to train ML/probabilistic models. The importance of user feedback to achieve high levels of quality is shared across all these works for specific tasks, e.g., repairing missing values. Nevertheless, the user has not been involved to manually clean data in a data cleaning process where data cleaning programs are based on data transformations, which is the most common approach in companies but not in research. Moreover, user involvement was not explored when continuous program refinements are applied through an iterative process. Note that CDC [52] addresses only stream data by proposing to specialize FDs, and Räth et al. [42] consider the user to verify the resulting data transformations.

General purpose tools. Customized data cleaning procedures can be implemented using a general purpose programming language, such as Java or Python, or a specialized software tool. Specialized commercial and research tools typically support different data operators that can be composed and form a transformation workflow to clean data. For instance, **ETL (Extraction, Transformation, and Loading)** tools such as Informatica PowerCenter[3], IBM Data Integration[4], Talend[5], and **Pentaho Data Integration (PDI)**[6] provide a panoply of data operators that can implement several data transformations on structured data. Data wrangling tools, such as Trifacta Wrangler[7] based on the research tool Data Wrangler [22], and OpenRefine[8], provide an interactive process

[3]https://www.informatica.com/products/data-integration/powercenter.html
[4]https://www.ibm.com/analytics/us/en/technology/data-integration/
[5]https://www.talend.com/
[6]http://community.pentaho.com/projects/data-integration/
[7]https://www.trifacta.com/
[8]http://openrefine.org/

to implement data transformations and to map data to a specific format through data operators. Finally, there are tools specifically designed for data cleaning purposes such as the commercial tool Trillium Quality[9] integrated in the Precisely Data Integrity Suite, and research tools.

Recent and general purpose data cleaning frameworks, LLUNATIC [18], NADEEF [11], and Horizon [44], aim at generating data repairs to address tuples violating data quality rules. Most recently, Garf [34] and Holoclean [43] provide automatic data repairs using probabilistic models. Both receive, from the user, the identification of a subset of clean data from a larger corpus. Holoclean also supports knowledge bases and **Denial Constraints (DCs)** as input. Older research tools such as FEBRL [9] and Ajax [16] support data transformations but, as ETL tools, do not support the user involvement when manually cleaning data.

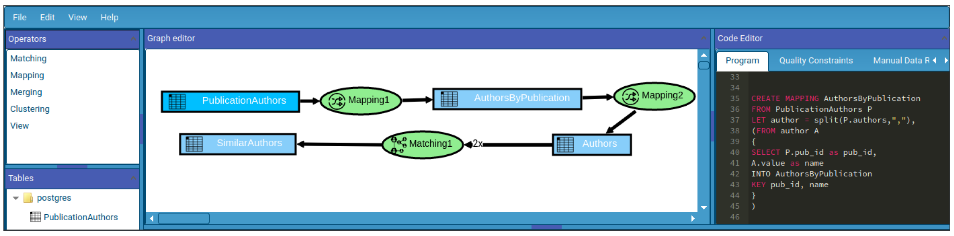
Large Language Models. Recently, with the popularity of **Large Language Models (LLMs)** that generate text (e.g., ChatGPT[10]) the automation of some data cleaning tasks is possible, namely, through the generation of code in a general purpose programming language or through prompt engineering, to transform a data cleaning task into a natural language question. Regarding the typical use cases of LLM, it can be used to (i) generate code for data transformations and (ii) perform data cleaning tasks that involve natural language analysis or correction, such as value imputation and approximate duplicate detection [31, 53]. In the first case, we foresee little adoption, as the code returned is usually a template that needs customization and which specialized tools already provide using intuitive graphical user interfaces. The usage (ii) can be complementary to specialized data cleaning frameworks and could be integrated in their pipelines, including also user involvement. Due to limitations in the input length of models like ChatGPT (e.g., 1,000 characters), high usage costs, and high latency, it becomes impractical to perform data cleaning tasks that involve processing medium to large amounts of data. Tasks such as resolving inconsistencies across tuples or tables or performing approximate duplicate detection for a large number of record pairs may not fit within the input size or could result in prolonged execution times.

Other research directions. Additional research directions for data cleaning, though not directly related with our current work, encompass: cleaning data with the (single) objective of improving ML models performance (CPClean [23] and AutoCure [1]), upgrading ML models fairness [47] or enhancing query results (Reference [5] and Dasy [19]), cleaning time series data [54], and integrating privacy issues into the data cleaning process [21].

4 CLEENEX

In this section, we describe Cleenex, which is based on the data cleaning framework called Ajax [16] that separates the logical specification of a data cleaning process from its physical implementation (SQL or Java algorithm implementations). The development of a data cleaning process in Cleenex involves the design of a data cleaning graph where the nodes are data operators or relational tables and the edges connect relational tables to data operators, as input or output. It supports five customizable data operators: (i) Map, which supports one-to-many operations that transform a tuple into one or more tuples (e.g., a split operation); (ii) Match, which computes an approximate join between two tables; (iii) Cluster, which groups the tuples of a table into partitions; (iv) Merge, which receives a table as input and outputs a tuple and its attributes for each partition accordingly to a given criteria algorithm; and (v) View, which supports the operations already present in the SQL language. Cleenex also supports a debugging mechanism that enables the user to select tuples from any table in the graph and navigate backwards or forwards through the data cleaning graph displaying the provenance of tuples.

[9]https://www.precisely.com/product/precisely-trillium/trillium-quality
[10]https://openai.com/research/chatgpt



```
CREATE MAPPING AuthorsByPublication
FROM PublicationAuthors P
LET author = split(P.authors,","),
(FROM author A
{
SELECT P.pub_id as pub_id,
A.value as name
INTO AuthorsByPublication
KEY pub_id, name
}
)
```

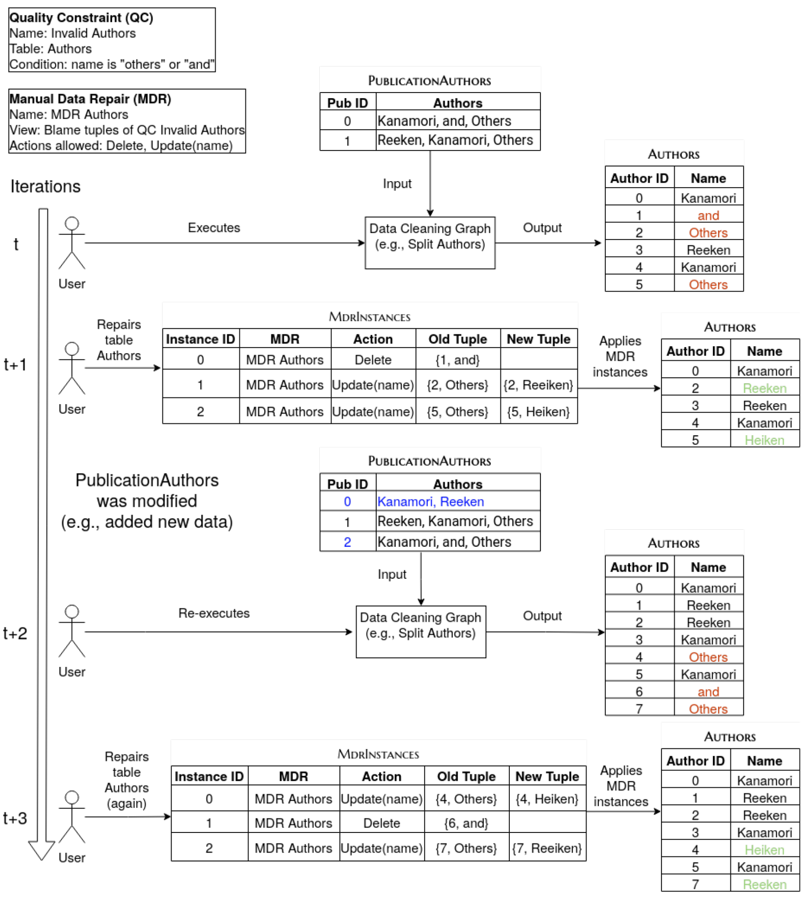
Fig. 5. Cleenex GUI with a data cleaning graph to detect approximate duplicate authors.

Example 4.1. Figure 5 presents the Cleenex GUI illustrating a data cleaning graph that performs the same data transformations of Example 2.1 (Section 2.1) to detect approximate duplicate authors from a list of publications (Table *PublicationAuthors*). The Splitting high-level data transformation is implemented in Cleenex with two Map operators: the first applies a splitting function for each publication, the second generates IDs for each author name. The Approximate Duplicate Detection is implemented with the Match operator that computes the edit-distance for each author name pair and returns the pairs whose distance value is less than 3.

To effectively support the user involvement, the designer may also specify Quality Constraints and/or Manual Data Repairs over any table of a data cleaning graph as proposed in Reference [17]. ***Quality Constraints* (QCs)** are integrity rules used to assess the data quality in a specific table of the graph. The tuples of a table that violate a quality constraint are named blamed tuples. Those blamed tuples bring the attention of the user to data tuples that do not satisfy QCs. ***Manual Data Repairs* (MDRs)** allow the user to manually modify data tuples that belong to tables in a data cleaning graph. In particular, MDRs can be used to manually correct blamed tuples. An MDR is defined by the set of allowed actions that can be taken over a table and a view. The actions can be: (i) updating a specific set of attribute values, (ii) deleting tuples, and/or (iii) adding tuples. The view mimics an SQL view and is used to specify the tuples and attributes of a relational table in the graph or the set of blamed tuples that are shown to the user. When the user manually repairs a data tuple, an MDR instance is created. An MDR instance stores the MDR name and the specific manual data modification performed by the user. An MDR instance is then used to apply the corresponding user manual data repair to the relational table when the data cleaning process re-executes.

Example 4.2 (User feedback in a data cleaning process). Consider the *PublicationAuthors* table that stores publication IDs and its author names as shown on the top of Figure 6. A User executes the data cleaning graph explained in Example 4.1. To simplify, we focus on the first data transformation, where we apply a *Splitting* transformation (using two Map operators in Cleenex) that splits the strings containing several author names stored in the *Authors* column of the *PublicationAuthors* table into several records and assigns an identifier to each name (*Author ID*). A QC to detect tuples whose author name contains either “Others” or “and” was previously specified. An MDR is defined with a view over the blamed tuples generated by violations of this QC and allows the user to delete or update the author name column value. The specifications of these QC and MDR are represented on top left of Figure 6. In Figure 6, we show the different iterations required to completely clean the data. The User executes this data cleaning process and manually repairs the incorrect author names that are either named “Others” or “and.” The iterative data cleaning process runs as follows:

— At iteration t, a User executes a data cleaning graph that, given the *PublicationAuthors* table as input, creates an *Authors* table where *Author ID* is the identifier of an author and *Name* is its name. The tuples 1, 2, and 5 of the *Authors* table are blamed tuples, because they violate the specified QC.



Quality Constraint (QC)
Name: Invalid Authors
Table: Authors
Condition: name is "others" or "and"

Manual Data Repair (MDR)
Name: MDR Authors
View: Blame tuples of QC Invalid Authors
Actions allowed: Delete, Update(name)

PublicationAuthors

Pub ID	Authors
0	Kanamori, and, Others
1	Reeken, Kanamori, Others

Authors

Author ID	Name
0	Kanamori
1	and
2	Others
3	Reeken
4	Kanamori
5	Others

MdrInstances

Instance ID	MDR	Action	Old Tuple	New Tuple
0	MDR Authors	Delete	{1, and}	
1	MDR Authors	Update(name)	{2, Others}	{2, Reeiken}
2	MDR Authors	Update(name)	{5, Others}	{5, Heiken}

Authors

Author ID	Name
0	Kanamori
2	Reeken
3	Reeken
4	Kanamori
5	Heiken

PublicationAuthors

Pub ID	Authors
0	Kanamori, Reeken
1	Reeken, Kanamori, Others
2	Kanamori, and, Others

Authors

Author ID	Name
0	Kanamori
1	Reeken
2	Reeken
3	Kanamori
4	Others
5	Kanamori
6	and
7	Others

MdrInstances

Instance ID	MDR	Action	Old Tuple	New Tuple
0	MDR Authors	Update(name)	{4, Others}	{4, Heiken}
1	MDR Authors	Delete	{6, and}	
2	MDR Authors	Update(name)	{7, Others}	{7, Reeiken}

Authors

Author ID	Name
0	Kanamori
1	Reeken
2	Reeken
3	Kanamori
4	Heiken
5	Kanamori
7	Reeken

Iterations
t
User
Executes
Input
Data Cleaning Graph (e.g., Split Authors)
Output
t+1
Repairs table Authors
User
Applies MDR instances
PublicationAuthors was modified (e.g., added new data)
Input
t+2
User
Re-executes
Data Cleaning Graph (e.g., Split Authors)
Output
t+3
Repairs table Authors (again)
User
Applies MDR instances

Fig. 6. Example of user feedback in Cleenex during a data cleaning process.

— At iteration $t+1$, using the MDR, the user manually repairs tuples 1, 2, and 5 in table *Authors*. MDR *Instance ID*=0 corresponds to deleting the tuple with the "and" value (tuple 1), MDRs *Instances ID*=1 *and 2* correspond to replace the "Others" by values "Reiiken" and "Heiken," respectively. The MDR instances are then applied to the Authors table, resulting in a clean table with no "Others" and "and" values.

— At iteration $t + 2$, the User re-executes the data cleaning graph, because the original *PublicationAuthors* table was modified (e.g., a new data record was added, *Pub ID*=0). For the purpose of illustration, consider that the publication ID generation is carried out externally (e.g., through a web crawler over semanticscholar.com), so not predictable (e.g., random). This time, the identifier of the publication with original *Pub ID*=0 is modified to *Pub ID*=2. The result is once again a dirty *Authors* table containing "Others" and "and" values in the name column.

— At iteration t+3, the User once again manually repairs the *Authors* table as she did in iteration $t + 1$, leading to a clean *Authors* table.

In Section 4.1, we focus on the Cleenex components that support the iterative execution of a data cleaning process. For a comprehensive architecture description and technical details, see Reference [35].

4.1 Supporting the Iterative Execution of a Data Cleaning Process

The execution of a data cleaning process is typically iterative mainly due to the following two cases: (i) the design of a data cleaning process is an iterative process where the designer refines the data cleaning graph, executes it, and then evaluates the quality of data produced; and (ii) the initial input data may change, e.g., when the amount of data is very large, it is more efficient to use a data sample (i.e., a small set of data) to design the data cleaning process. After the process is refined, the designer executes the data cleaning process to the whole dataset. Due to those two cases, Cleenex may produce new data in different points of the graph, e.g., the new *PublicationAuthors* table in iteration $t+2$ of Figure 6. Every time a data cleaning process reruns, Cleenex automatically reapplies the actions contained in MDR instances produced in a previous run to the relational data tables in the graph; otherwise, the user has to perform those actions again as in iteration $t + 3$ of Figure 6.

There are two problems with the automatic reapplication of MDR instances. First, the MDR instances do not persist in disk, so they are lost when a program re-executes. Second, the reapplication of MDR instances to modified and/or additional data as explained above is a challenge, because the data tuples may not hold all their original attribute values. We define that an *MDR instance conflict* arises when Cleenex cannot automatically apply an MDR instance due to considerable data changes that do not guarantee that the modifications still hold. For example, in iteration $t + 2$ of Figure 6, the incorrect author names, i.e., "and" and "Others" values, have now different *Author ID* values, so Cleenex is not able to reapply the MDR instance to the right tuple.

To support the iterative execution of a data cleaning process with the above constraints, we first added to Cleenex the *MDR Persistence* component that prevents the loss of previous MDR instances when the data cleaning process is re-executed with the same data. Then, we created the *MDR Recovery* component that tries to reapply MDR instances in case of conflict or asks the user in a concise way for additional feedback. To develop such component, in Reference [33], the author introduced the notion of deterministic attributes in a Cleenex data cleaning graph specification. *Deterministic attributes* are attributes whose values do not change during the iterative execution of a data cleaning process (e.g., column *Name* of table *Authors* in Figure 6). So, only the values of deterministic attributes are taken into account to reapply MDR instances, while the remaining values are ignored because they may have changed. In the context of this component, we formulated and implemented an algorithm (detailed in Section 4.1.1) for the detection and automatic recovery of MDR instances,[11] which tries as much as possible with 100% guarantees to recover the MDR instances from a conflict. If it is not possible to recover, then the algorithm calls the user to help selecting the right data tuples to which the MDR instance should be reapplied.

Example 4.3 (Iterative data cleaning process). To exemplify the usefulness of the MDR Persistence and the MDR Recovery components, we revisit the example of Figure 6—defining column *Name* of table *Authors* as deterministic and recreating iterations $t + 2$ and $t + 3$ in Figure 7 using these two components:

— At iteration $t + 2$, the generated dirty *Authors* table is passed as input to the MDR Recovery component. The MDR Recovery component obtains the MDR instances created in iteration $t + 1$ through the MDR Persistence component, which retrieves them from the database. For *Instance ID*=0, it can recover the tuple, because there is only one "and" and it is placed in a deterministic column, so it removes the tuple. Since there are two tuples with "Others," the component is not able to identify which MDR instance to apply to each tuple, so it creates a

[11]The thesis cited in Reference [33] proposed the first version of the algorithm and developed the GUI to support conflict resolution.

ALGORITHM 1: MDR instance recovery and conflict resolution algorithm

```
Input:
view = view specified in the MDR, includes the tuples in the view;
mdr_instances = list of MDR instances that belong to the same MDR;
pk_atts = list of attributes that constitute the primary key of the base relation of the view vr;
det_atts = list of attributes of the base relation of the view vr that have been declared as deterministic;
Output: conflicts = list of MDR instance conflicts
1  conflicts ⟵ ∅;
2  foreach inst ∈ mdr_instances do
3      if inst.action = insert then view ⟵ view ∪ inst.new_tuple ; // There are no MDR Conflicts
4      else if pk_atts ⊆ det_atts then applyMDRInstance (inst, pk_atts, view) ; // There are no MDR Conflicts
5      else
6          candidate_tuples ⟵ {tuple ∈ view such that tuple.att = inst.old_tuple.att for all att ∈ det_atts};
7          if |candidate_tuples| = 1 then applyMDRInstance (inst, det_atts, view) ; // There are no MDR Conflicts
8          else
9              new_conflict ⟵ conflict (inst, candidate_tuples);
10             conflicts ⟵ conflicts ∪ new_conflict;
11 return conflicts;
   /* Applies the MDR instance to every tuple that for all attributes passed as argument, the values are equal
      to the instance old tuple values                                                                    */
12 Procedure applyMDRInstance (inst, atts, view):
13     foreach tuple ∈ view do
14         if tuple.att = inst.old_tuple.att for all att ∈ atts then
15             if inst.action = delete then view ⟵ view \tuple;
16             else // Update MDR
17                 new_tuple ⟵ tuple;
18                 foreach att ∈ inst.action.update do new_tuple.att ⟵ inst.new_tuple.att ;
19                 view ⟵ view \tuple ∪ new_tuple;
```

conflict for MDR instances 1 and 2 saving the candidate tuples **Primary Keys (PK)** values (in this example, the Primary Key is the *Author ID* column).

— At iteration $t + 3$, the User selects one conflict to resolve. She selects the MDR *Instance ID*=1 and selects the tuple with *Author ID*=7 to which this MDR instance should apply. The MDR instance is updated with the new tuple and it is applied to the *Authors* table, leaving only one tuple with the value "Others." The MDR recovery component is called with the new MDR instances and the new *Author* table. Now, because only one tuple contains the "Others" value, the MDR recovery component is able to assign it to MDR *Instance ID*=2, then updates the MDR instance and applies it to the *Authors* table, resulting in a cleaned table. Note that, although the number of iterations is the same, the User only had to select the tuple to which an MDR instance should be applied while before, as illustrated in Figure 6, the User had to produce all the MDR instances from scratch.

4.1.1 Conflict Detection and Automatic Recovery Algorithm. We present the algorithm for the conflict detection and automatic recovery of MDR instances in Algorithm 1. This algorithm receives a list of MDR instances that belong to the same MDR, applies the MDR instances that have no conflicts, and outputs a list of MDR instance conflicts to be resolved by the user.

At each new execution of the data cleaning graph, the algorithm first detects existing MDR instance conflicts by evaluating each MDR instance (line 2). If an MDR instance contains an insert action, then there is no conflict possible, and it is immediately applied (line 3). If all attributes that form the Primary Key involved in the MDR instance are deterministic, then there is also no conflict, and the MDR instance can be applied (line 4). For each remaining MDR instances, the algorithm checks the tuples whose deterministic attribute values are equal to the MDR instance Old Tuple values (line 6). If there is only one tuple, then the algorithm can apply the MDR

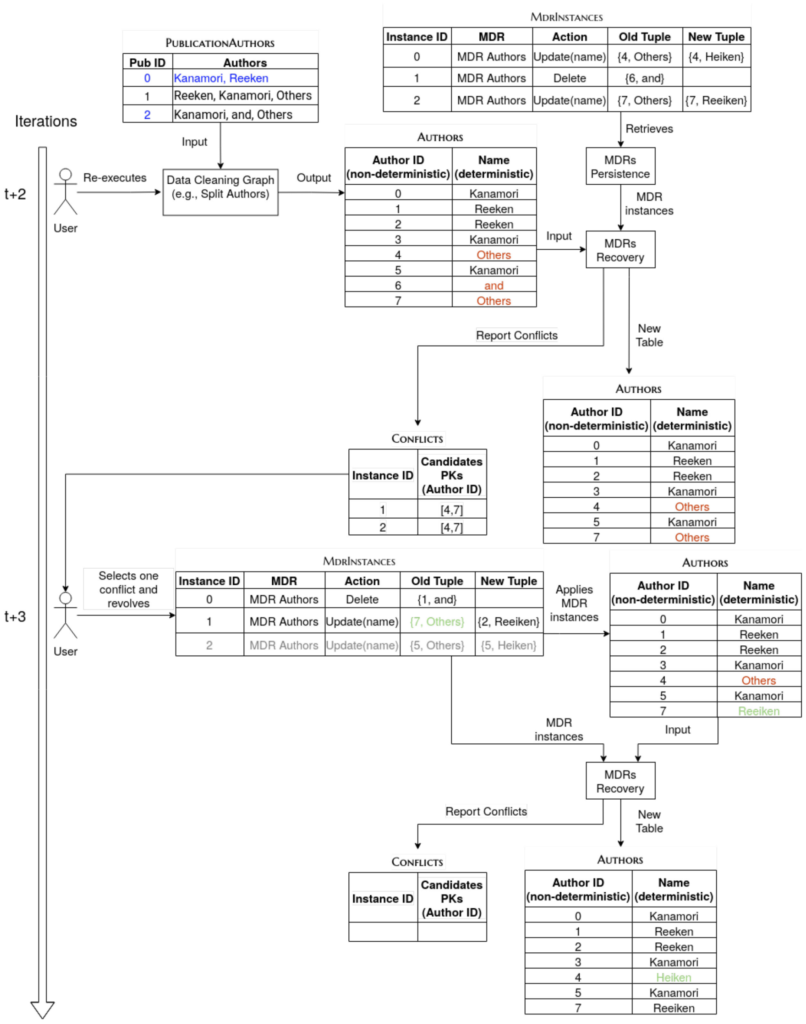

PublicationAuthors
Pub ID | Authors
0 | Kanamori, Reeken
1 | Reeken, Kanamori, Others
2 | Kanamori, and, Others
Iterations
t+2
t+3
User
Re-executes
Input
Data Cleaning Graph (e.g., Split Authors)
Output
MdrInstances
Instance ID | MDR | Action | Old Tuple | New Tuple
0 | MDR Authors | Update(name) | {4, Others} | {4, Heiken}
1 | MDR Authors | Delete | {6, and} |
2 | MDR Authors | Update(name) | {7, Others} | {7, Reeiken}
Retrieves
MDRs Persistence
MDR instances
MDRs Recovery
Authors
Author ID (non-deterministic) | Name (deterministic)
0 Kanamori
1 Reeken
2 Reeken
3 Kanamori
4 Others
5 Kanamori
6 and
7 Others
Report Conflicts
New Table
Conflicts
Instance ID | Candidates PKs (Author ID)
1 | [4,7]
2 | [4,7]
Selects one conflict and revolves
MdrInstances
0 | MDR Authors | Delete | {1, and} |
1 | MDR Authors | Update(name) | {7, Others} | {2, Reeiken}
2 | MDR Authors | Update(name) | {5, Others} | {5, Heiken}
Applies MDR instances
7 Reeiken
4 Heiken
7 Reeiken

Fig. 7. Example of user feedback in Cleenex using the MDR persistence and the MDR recovery components during an iterative data cleaning process.

instance (line 7), e.g., deleting "and" in Figure 7 iteration $t+2$ and updating the remaining "Others" value in Figure 7 iteration $t+3$; otherwise, a new conflict has to be created (lines 9–10), e.g., the two "Others" values in Figure 7 iteration $t + 2$. A conflict contains the original MDR instance and the tuples found whose deterministic values are equal to the MDR instance Old Tuple values. When the algorithm finishes, the user can inspect the conflicts returned by the algorithm and decide how to resolve them.

5 USER INVOLVEMENT EVALUATION

We conducted an extensive experimental validation to evaluate the support for the user involvement in Cleenex with two studies: (i) with a simulated user, we evaluated the Cleenex components regarding their support for the user feedback during a data cleaning process; and (ii) with real users, we conducted an experimental evaluation of user involvement during a data cleaning process with Cleenex against two data preparation tools typically used for data cleaning: OpenRefine and PDI. In Reference [36], we had already conducted a preliminary experimental study to compare Cleenex against OpenRefine and PDI in the context of Approximate Duplicate Detection and Consolidation.

We report the research questions, experimental setup, and obtained results to evaluate Cleenex support for the user feedback in Section 5.1. We detail our experiments with real users on Cleenex, OpenRefine, and PDI in Section 5.2.

5.1 User Feedback Support in Cleenex

The goal of this study was to evaluate the effectiveness of the different components incorporated in Cleenex (namely, QC/MDR Managers, MDRs persistence, and MDRs recovery) in terms of their effect for reducing the user effort when manually cleaning data during a data cleaning process. **Research Questions.** Specifically, we investigate the following *research questions*:

- **RQ1.1:** What is the impact of having QCs and MDRs in the context of a data cleaning process?
- **RQ1.2:** What is the impact of having persistent MDR instances in the context of an iterative data cleaning process?
- **RQ1.3:** What is the impact of having MDR conflict resolution in the context of an iterative data cleaning process?

Experimental Setup. To answer those research questions, we programmed an ideal user to execute a data cleaning program, refined by a designer, and to manually clean the required data during a data cleaning process to obtain 100% clean data. We used the following two datasets: (i) **Big Publication Authors (BPA)**, which consists of a table that lists publications and corresponding authors where some author names are written in similar but different ways (approximate duplicates); and (ii) **Customers (C)**, which contains three tables: one for customers with inconsistent phone number formats, incorrect/missing values in customer name, phone number, street and city, another table for treatments where insurance names are written differently (approximate duplicates), and a master data table that contains a cleaned subset of customer records.

For each dataset, we measured the user effort needed to clean the data quality problems left after executing a data cleaning program designed by an expert: the number of tuples and characters (**chars**) visualized, inserted, updated, and deleted by the user to obtain 100% clean data.
Results. Table 2 reports the results after executing the first iteration of the data cleaning program for the BPA dataset using Cleenex with QCs and MDRs. We compare it against the baseline, which is the alternative way available for the user to manually clean the data and obtain equal end results without QCs and MDRs. In Table 3, we report further iterations over the same data cleaning process for BPA with QCs and MDRs. We measured the user effort when cleaning again the same data quality problems with the MDRs Persistence component and with and without the MDRs Recovery component. We compare those two components against the baseline, which is to use Cleenex without such components where only QCs and MDRs are available but MDR instances are lost after each iteration. Table 4 reports the results of executing the data cleaning program for the C dataset with and without QCs and MDRs.

- **RQ1.1.** Based on the results reported in Tables 2 and 4, we observe that performing a data cleaning process with QCs/MDRs reduces the user effort, because the user needs to visualize less data (74%–99% less tuples and 86%–99% less chars) and perform effortless manual data repairs (64%–94% less chars inserted/updated) comparatively to the same process without QCs/MDRs.

Table 2. User Results after Executing the First Iteration of the Data Cleaning Process for the BPA Dataset. The gain percentage with respect to the first column values is reported in parentheses in the last column

		Without Qcs/MDRs (Baseline)	With Qcs/MDRs
Visualization	# tuples visualized	1,413	**365** (74%)
	# chars visualized	51,259	**7,308** (86%)
Insertion	# tuples inserted	214	**0** (100%)
	# chars inserted	2,589	**0** (100%)
Update	# tuples updated	**0**	100 (N/A)
	# chars updated	**0**	939 (N/A)
Deletion	# tuples deleted	**1**	165 (-16400%)
Total	# tuples inserted/updated/deleted	**215**	265 (-23%)
	# chars inserted/updated	2,589	**939** (64%)

Table 3. User Results per Iteration (Excluding the First One) of the Iterative Data Cleaning Process for the BPA Dataset. The gain percentage with respect to each of the two left columns values is reported in parentheses in the adjacent column to right

		Without MDRs Persistence and MDRs Recovery (Baseline)	With MDRs Persistence and without MDRs Recovery	With MDRs Persistence and MDRs Recovery
Visualization	# tuples visualized	365	265 (27%)	**106** (60%)
	# chars visualized	7,308	6,426 (12%)	**2,481** (61%)
Insertion	# tuples inserted	**0**	**0** (N/A)	**0** (N/A)
	# chars inserted	**0**	**0** (N/A)	**0** (N/A)
Update	# tuples updated	100	**0** (100%)	**0** (N/A)
	# chars updated	939	**0** (100%)	**0** (N/A)
Deletion	# tuples deleted	165	165 (0%)	**26** (84%)
Total	# tuples inserted/updated/deleted	265	165 (38%)	**26** (84%)
	# chars inserted/updated	939	**0** (100%)	**0** (N/A)

Table 4. User Results after Executing the Data Cleaning Process for the C Dataset. The gain percentage with respect to the first column values is reported in parentheses in the last column

		Without Qcs/MDRs (Baseline)	With Qcs/MDRs
Visualization	# tuples visualized	176,200	**1,739** (99%)
	# chars visualized	9,902,194	**145,017** (99%)
Insertion	# tuples inserted	**0**	**0** (N/A)
	# chars inserted	**0**	**0** (N/A)
Update	# tuples updated	268	**69** (74%)
	# chars updated	5,851	**333** (94%)
Deletion	# tuples deleted	**0**	200 (N/A)
Total	# tuples inserted/updated/deleted	**268**	269 (0%)
	# chars inserted/updated	5,851	**333** (94%)

RQ1.2. The MDRs Persistence component reduced the user effort when manually cleaning data in an iterative data cleaning process. As observed in Table 3, the number of tuple updates is reduced to zero and the number of tuples visualized dropped 27%.

RQ1.3. Regarding an iterative data cleaning process with the MDRs Recovery component, we observe also in Table 3 that this component reduces the user effort. The MDRs Recovery component was able to recover most of the manual data repair instances created by the user in previous iterations of the data cleaning process. For the cases where recovery was not possible, the MDRs Recovery component guided the user into recovering conflicting MDR instances from previous iterations with less effort (84% less deleted tuples) than creating those MDR instances again.

5.2 User Involvement Support in a Data Cleaning Process

In this study, we aimed at evaluating whether Cleenex effectively helps program designers and domain expert users to perform data cleaning tasks in a realistic scenario. We investigated different aspects of usability of Cleenex, OpenRefine, and PDI.

Research Questions. Specifically, we try to answer the following *research questions*:

- **RQ2.1:** Are Cleenex data cleaning programs easier to understand than programs specified using the other two tools?
- **RQ2.2:** Given a specification of a data cleaning program written and refined by an expert, does using Cleenex require less time/user effort to obtain cleaned data than using the other two tools?
- **RQ2.3:** Given a specification of a data cleaning program written and refined by an expert, does the data obtained with Cleenex have higher quality than the data obtained with the other two tools?
- **RQ2.4:** Does the debugging functionality help to obtain more correct data than the other two tools?

Experimental Setup. We performed a controlled experiment with expert users, using a within-subject design in which all participants from each group tested the two tools under comparison in that group. The preparation of this study included: (i) a video tutorial for each tool, (ii) an experimental plan, (iii) an experimental guide, and (iv) a satisfaction questionnaire. We used satisfaction questionnaires to collect users' opinions instead of the think-aloud method, because typically it affects the users' performance, and we wanted to measure the task completion time. We asked the users to fill a form with demographic information (age, degree, and details about their current occupation), watch the corresponding video tutorial of their first tool, then to perform two tasks and answer a set of questions after performing these tasks to assess their perception. This process was repeated for the second tool and ended with questions to assess and to detail their preferences. The satisfaction questionnaire included: (i) questions about the user perception, as listed in Table 5, distinguished by the Research Question identifier they are intended to help answering or classified as Overall if it is a general question about user preference; and (ii) the questions from the standard **Technology Acceptance Model (TAM)** questionnaire [12] as listed in Table 6. TAM questions are grouped into usefulness and ease of use. The evaluation of these TAM questions is usually performed by analyzing the sum of the scores obtained for each group.

Experimental Task Descriptions. The first task, *Task 1*, contains several questions that evaluate the user's understandability regarding one of the data cleaning programs designed by an expert. The programs were similar in complexity (i.e., similar amount and type of transformations) and included typical data transformation tasks in data cleaning that implement: (i) an approximate duplicate detection and consolidation process; and (ii) that produce additional rows or columns based on column string values. We evaluated the score obtained for each understandability question to measure how easy it is to understand a program. The second task, *Task 2*, uses the same data cleaning program and requires the user to manually clean data to obtain 100% clean data. So, we measured the final output data quality in terms of Precision, Recall, and F1-Measure. During the execution of Task 1 and Task 2, we measured the number of clicks and keys pressed by the user, as well as the time to complete each task and the number of times the Cleenex debugger was used. Users had to perform both tasks using Cleenex and using another tool. We used two datasets with the corresponding data cleaning program: (i) **Childhood Locations (CL)**, based on a real dataset that lists Chicago early childhood locations that contains approximate duplicate records; and (ii) **Publication Authors (PA)**, which consists of a table that lists publications and corresponding authors where some author names are written in similar but different ways (approximate duplicates).

Table 5. Perception Questions Asked to the User

Group	ID	Question	Type/Scale	Research Question
User Perception (after completing Task 1)	UP1	How easy was it to understand the data cleaning program?	1- very difficult, 7- very easy	RQ2.1
	UP2	How difficult was it to answer the questions about the data cleaning program?	1- very difficult, 7- very easy	RQ2.1
	UP3	How much effort was required to answer the questions about the data cleaning program?	1- almost no effort, 7 – extreme effort	RQ2.1
Program Specification	PS1	It was easy to understand this tool operators	1 - strongly disagree, 7 – strongly agree	RQ2.1
	PS2	It was easy to understand the source of specific records using this tool	1 - strongly disagree, 7 – strongly agree	RQ2.3
Manually correcting data	MCD1	It was easy to find the records that I wanted to modify using [this tool].	1 - strongly disagree, 7 – strongly agree	RQ2.2
	MCD2	It was easy to manually repair records using [this tool].	1 - strongly disagree, 7 – strongly agree	RQ2.2
User Preference	UPF1	What would you use to specify a data cleaning program?	1 – PDI/OpenRefine, 7 – Cleenex	Overall
	UPF2	Please describe the main reasons for your answer above.	Open answer	Overall
	UPF3	Consider that you have the same data cleaning program implemented in both tools, which tool would you use to understand the source of specific records in a data cleaning program?	1 – PDI/OpenRefine, 7 – Cleenex	Overall
	UPF4	Please describe the main reasons for your answer above.	Open answer	Overall
	UPF5	Consider that you have the same data cleaning program implemented in both tools, what would you use to manually modify data?	1 – PDI/OpenRefine, 7 – Cleenex	Overall
	UPF6	Please describe the main reasons for your answer above.	Open answer	Overall
	UPF7	Based on your experience with Cleenex, please describe which additional feature(s) would you like to see implemented and why (you can compare with other tools)?	Open answer	Overall

Table 6. TAM Satisfaction Questionnaire Questions Asked to the User

Group	Question	Scale
Usefulness	Using this tool to perform data cleaning tasks would enable me to accomplish tasks more quickly.	1 - strongly disagree, 7 – strongly agree
	Using this tool to perform data cleaning tasks would improve my performance (quality of output).	1 - strongly disagree, 7 – strongly agree
	Using this tool to perform data cleaning tasks would increase my productivity (efficiency of production, Output/Input).	1 - strongly disagree, 7 – strongly agree
	Using this tool would enhance my effectiveness (accomplish to do the right tasks) to perform data cleaning tasks.	1 - strongly disagree, 7 – strongly agree
	Using this tool would make it easier to perform data cleaning tasks.	1 - strongly disagree, 7 – strongly agree
	I would find this tool useful to perform data cleaning tasks.	1 - strongly disagree, 7 – strongly agree
Ease of use	Learning to operate with this tool would be easy for me.	1 - strongly disagree, 7 – strongly agree
	I would find it easy to get this tool to do what I want it to do.	1 - strongly disagree, 7 – strongly agree
	My interaction with this tool would be clear and understandable.	1 - strongly disagree, 7 – strongly agree
	I would find this tool to be flexible to interact with.	1 - strongly disagree, 7 – strongly agree
	It would be easy for me to become skillful at using this tool.	1 - strongly disagree, 7 – strongly agree
	I would find this tool easy to use.	1 - strongly disagree, 7 – strongly agree

The programs executed by the users that experimented with PDI were smaller, because PDI does not support approximate duplicate consolation natively, i.e., without programming an operator.

Participants Characterization and Grouping. There were three groups of users: (i) users who performed experiments on Cleenex and OpenRefine, (ii) users who performed experiments on Cleenex and PDI, and (iii) users who typically use PDI in their work and performed the experiments using Cleenex and PDI. The regular participants (user groups (i) and (ii)) in this study were 24 (12 in each group) and the PDI experts (user group (iii)) were 8 in total. Half users in group (i) and (ii) started the session with Cleenex, while the other half started with OpenRefine or PDI. This counterbalance is the recommended method to neutralize the learning effect that exists in within-subjects experiments (for additional details, please see Section 3.5 from Reference [26] and Section 14.2.2 from Reference [48]). The age range for all groups was 21–45, mean around 28–32 for each group. Twelve users in (i) and (ii) were students who took a data integration course with theoretical and lab experience on data cleaning tasks; while the other 12 performed data cleaning tasks in their jobs, both were evenly distributed. In terms of the highest level of education completed, 11 users

Table 7. Metrics and Answers Related to RQ2.1, P-values <= 0.05, which Pass the Statistical Test, Are Bolded

			Cleenex vs. OpenRefine			Cleenex vs. PDI			Experts – PDI vs. Cleenex		
Measure Type	**Tools**	**Measures**	**AVG**	**MEDIAN**	**P-value**	**AVG**	**MEDIAN**	**P-value**	**AVG**	**MEDIAN**	**P-value**
Task1 - Understandability	Cleenex	Task 1 # Correct Answers	7.67	8.00	**1.25E-04**	6.92	7.00	**1.21E-06**	6.88	7.00	**1.05E-06**
	Other Tool	Task 1 # Correct Answers	5.92	6.00	5.75E-01	5.00	5.00	5.98E-01	6.50	7.00	**6.32E-05**
	Tools Difference	Task 1 # Correct Answers	1.75	2.00	**3.91E-03**	1.92	2.00	**9.77E-04**	0.38	0.00	1.97E-01
User Effort	Cleenex	Task1 # Clicks	57.92	38.00	**5.76E-03**	39.58	36.50	7.11E-01	58.88	53.00	1.83E-01
	Other Tool	Task1 # Clicks	117.08	78.00	**3.78E-04**	171.00	154.00	7.15E-01	116.38	85.00	**4.61E-03**
	Tools Difference	Task1 # Clicks	−59.17	−29.00	**3.27E-02**	−131.42	−120.50	**4.88E-04**	−34.38	−37.00	1.48E-01
	Cleenex	Task1 # Keys	0.00	0.00	**1.00E-05**	0.00	0.00	**1.00E-05**	0.38	0.00	**4.79E-04**
	Other Tool	Task1 # Keys	5.92	0.00	**2.78E-05**	3.83	1.00	**1.06E-04**	2.38	0.00	**1.52E-05**
	Tools Difference	Task1 # Keys	−5.92	0.00	1.54E-01	−3.83	−1.00	9.76E-02	0.25	0.00	3.57E-01
User Perception (after completing the task)	Cleenex	Answers to UP1	5.67	6.00	**2.28E-02**	5.83	6.00	**2.90E-03**	5.38	5.00	3.24E-01
	Other Tool	Answers to UP1	4.08	4.00	2.11E-01	5.58	6.00	5.88E-02	5.00	5.00	1.20E-01
	Tools Difference	Answers to UP1	1.58	1.50	**3.91E-03**	0.25	0.00	5.63E-01	0.38	0.50	5.31E-01
	Cleenex	Answers to UP2	5.58	5.50	1.18E-01	5.42	6.00	**2.81E-02**	5.88	6.00	**3.70E-02**
	Other Tool	Answers to UP2	3.42	3.00	**1.52E-02**	4.00	4.00	2.59E-01	5.13	5.00	1.95E-01
	Tools Difference	Answers to UP2	2.17	2.00	**4.88E-04**	1.42	1.50	6.54E-02	0.75	1.00	2.19E-01
	Cleenex	Answers to UP3	3.00	2.50	6.57E-02	2.92	2.50	**1.95E-02**	2.63	2.50	2.45E-01
	Other Tool	Answers to UP3	4.50	5.00	**1.73E-02**	4.33	5.00	5.89E-01	3.25	3.00	4.08E-01
	Tools Difference	Answers to UP3	−1.50	−1.50	**3.81E-02**	−1.42	−1.50	**3.13E-02**	−0.63	−0.50	3.13E-01
Easiness User Perception	Cleenex	Answers to PS1	5.25	6.00	**2.72E-02**	6.08	6.00	**1.52E-02**	5.25	5.50	**1.85E-02**
	Other Tool	Answers to PS1	3.42	3.50	1.89E-01	4.92	5.50	9.98E-02	5.13	5.50	**3.56E-02**
	Tools Difference	Answers to PS1	1.83	1.50	**4.10E-02**	1.17	0.50	9.38E-02	0.13	0.00	7.85E-01
	Cleenex	Answers to PS2	5.67	6.00	**2.28E-02**	6.33	6.50	5.21E-03	6.63	7.00	**4.79E-04**
	Other Tool	Answers to PS2	3.25	3.00	4.95E-01	4.58	5.50	5.88E-02	5.25	6.00	7.45E-02
	Tools Difference	Answers to PS2	2.42	2.00	**9.77E-04**	1.75	1.50	**4.69E-02**	1.38	1.00	6.25E-02

had a bachelor's degree, 12 a master's degree, and 1 a PhD. As far as we know, this is the first study reaching such a number of representative participants (32 in total) to perform data cleaning tasks.
Results. We organized the different results per research question. We report the results for RQ2.1, RQ2.2, RQ2.3, and RQ2.4, respectively, in Tables 7, 8, 9, and 10. In addition, we report overall results regarding the TAM scores and user preferences in Table 11. For each tool and for each question score and measure, we report the average, the median of the values, and the P-value. The P-value corresponds to a normality test, namely, Shapiro-Wilk Test indicated for small sample size (<50). We consider that the variable (i.e., the results for a given tool) follows a normal distribution if P-value <= 0.05. For each question score and measure, when possible, we also report the difference between the values obtained for Cleenex and for another tool within the rows labeled "Tools Difference" in the "Tools" column. For the difference, we report the average, the median, and the P-value that corresponds to a pair test over the values obtained on both tools. If both testing variables (measure values obtained with Cleenex and another tool) are normally distributed, then we applied the Paired t-test, otherwise, we applied the Wilcoxon Test. We statistically accept that Cleenex is better in a particular measure or question if the test passes with P-value <= 0.05.

RQ2.1. As observed in Table 7, Cleenex did not pass the statistical test against OpenRefine in the number of keys pressed for Task 1. The number of correct answers for Task 1, the scores for all user perception questions (UP1, UP2, UP3, PS1, and PS2 listed in Table 5), and the number of clicks for Task 1 passed the test. Regarding Cleenex against PDI, the number of correct answers for Task 1, the scores for questions UP3 and PS2, and the number of clicks for Task 1 passed the test. In the experiments with PDI experts, we could not statistically prove any metric, because the P-values of the Tools Differences are higher than 0.05. Nevertheless, all users, including both regular participants and PDI experts, performed similar or better on Cleenex and answered the questions positively to Cleenex against OpenRefine and PDI.

RQ2.2. In Table 8, we observe that almost all metrics in all types of experiments (Cleenex vs. OpenRefine, Cleenex vs. PDI, and PDI experts) are statistically proved. The exception is the number of Keys pressed for Task 2 in the experiments of Cleenex vs. PDI and PDI

Table 8. Metrics and Answers Related to RQ2.2, P-values <= 0.05, which Pass the Statistical Test, Are Bolded

			Cleenex vs. OpenRefine			Cleenex vs. PDI			Experts – PDI vs. Cleenex		
Measure Type	**Tools**	**Measures**	**AVG**	**MEDIAN**	**P-value**	**AVG**	**MEDIAN**	**P-value**	**AVG**	**MEDIAN**	**P-value**
User Effort	Cleenex	Task2 Execution Time (s)	636.50	576.50	9.44E-01	248.83	199.50	**1.70E-03**	234.00	208.50	**2.61E-02**
	Other Tool	Task2 Execution Time (s)	850.42	834.00	9.28E-01	508.75	423.50	**9.42E-03**	586.88	526.00	**4.38E-02**
	Tools Difference	Task2 Execution Time (s)	−180.50	−142.00	**4.88E-04**	−259.92	−296.00	**1.72E-02**	−352.88	−336.00	**1.99E-02**
	Cleenex	Task2 # Clicks	52.08	44.00	**3.37E-02**	14.67	11.50	**4.99E-02**	21.38	21.50	3.56E-01
	Other Tool	Task2 # Clicks	113.50	101.00	1.06E-01	72.42	71.50	2.68E-01	116.38	85.00	**4.61E-03**
	Tools Difference	Task2 # Clicks	−56.58	−47.00	**1.46E-03**	−57.75	−58.50	**1.95E-03**	−95.00	−63.50	**7.81E-03**
	Cleenex	Task2 # Keys	0.17	0.00	**1.21E-06**	0.00	0.00	**1.00E-05**	0.00	0.00	**1.00E-05**
	Other Tool	Task2 # Keys	14.67	6.00	**9.62E-04**	28.17	1.00	**1.52E-04**	2.38	0.00	**1.52E-05**
	Tools Difference	Task2 # Keys	−14.17	−6.00	**3.25E-02**	−28.17	−1.00	9.67E-02	−2.38	0.00	2.70E-01
User Effort Perception	Cleenex	Answers to MCD1	5.58	6.00	**2.80E-02**	6.33	7.00	**1.05E-03**	6.25	6.50	**1.85E-02**
	Other Tool	Answers to MCD1	3.17	3.50	6.05E-02	3.92	4.00	5.50E-01	3.63	3.50	1.62E-01
	Tools Difference	Answers to MCD1	2.42	2.50	**9.77E-03**	2.42	2.50	**5.86E-03**	2.63	3.00	**1.56E-02**
	Cleenex	Answers to MCD2	5.25	6.00	**8.92E-03**	6.50	7.00	**1.69E-04**	7.00	7.00	**1.00E-05**
	Other Tool	Answers to MCD2	3.17	3.00	8.69E-02	3.75	3.50	8.93E-01	5.25	5.50	3.34E-01
	Tools Difference	Answers to MCD2	2.08	2.00	**2.88E-02**	2.75	3.00	**3.91E-03**	1.75	1.50	**3.13E-02**

Table 9. Metrics and Answers Related to RQ2.3, P-values <= 0.05, which Pass the Statistical Test, Are Bolded

			Cleenex vs. OpenRefine			Cleenex vs. PDI			Experts – PDI vs. Cleenex		
Measure Type	**Tools**	**Measures**	**AVG**	**MEDIAN**	**P-value**	**AVG**	**MEDIAN**	**P-value**	**AVG**	**MEDIAN**	**P-value**
Task 2 – Output Quality	Cleenex	Precision	0.83	1.00	**9.81E-06**	1.00	1.00	**1.00E-05**	1.00	1.00	**1.00E-05**
	Other Tool	Precision	0.51	0.50	**3.52E-02**	0.94	1.00	**4.40E-05**	1.00	1.00	**1.00E-05**
	Tools Difference	Precision	0.32	0.25	**1.06E-02**	0.06	0.00	8.19E-02	0.00	0.00	1.00E+00
	Cleenex	Recall	0.79	1.00	**6.71E-05**	1.00	1.00	**1.00E-05**	1.00	1.00	**1.00E-05**
	Other Tool	Recall	0.71	1.00	**2.34E-04**	0.94	1.00	**1.21E-06**	0.96	1.00	**1.05E-06**
	Tools Difference	Recall	0.08	0.00	1.66E-01	0.06	0.00	3.39E-01	0.04	0.00	3.51E-01
	Cleenex	F1-Measure	0.81	1.00	**4.55E-05**	1.00	1.00	**1.00E-05**	1.00	1.00	**1.00E-05**
	Other Tool	F1-Measure	0.58	0.67	**2.80E-02**	0.92	1.00	**9.77E-05**	0.98	1.00	**1.05E-06**
	Tools Difference	F1-Measure	0.23	0.17	**1.18E-02**	0.08	0.00	9.65E-02	0.03	0.00	3.51E-01

experts that did not pass the test. Still, participants on Cleenex pressed less keys than in other tool.

RQ2.3. Table 9 presents the output data quality resulting from Task 2. We statistically prove the results of Precision and F1-Measure for Cleenex against OpenRefine. While no other statistical significance was observed regarding RQ2.3, the participants using Cleenex achieved better quality than using another tool, even among the PDI experts.

RQ2.4. We observe, in Table 10, that the debugger was used on Task 1 but was never used on Task 2. Regarding answers to question PS2, listed in Table 5, we verify with statistically significance that users find it easier to discover the source of a record using Cleenex when compared to OpenRefine and PDI. Among PDI experts, we also got a better PS2 but without statistical proof. Moreover, as observed in the answers to question UPF3, all participant groups preferred Cleenex to other tool.

Based on the obtained results:

— We concluded that: (i) Cleenex data cleaning programs are easier to understand for regular users than programs specified using the other considered tools; (ii) given a specification of a data cleaning program written and refined by an expert, Cleenex requires less time/user effort to obtain cleaned data than any of the other two tools; and (iii) the debugging functionality helps to understand data cleaning programs.

— Still, with no statistical proof but with positive results, we observed that given a specification of a data cleaning program written and refined by an expert, the data obtained with Cleenex has higher quality than using any of the other two tools.

— As observed in Table 11, the user answer values to the TAM [12] questionnaire and to the quantitative Overall questions on User Preference (UPF1, UPF3, and UPF5 listed in Table 5)

Table 10. Metrics and Answers Related to RQ2.4, P-values <= 0.05, which Pass the Statistical Test, Are Bolded

			Cleenex vs. OpenRefine			Cleenex vs. PDI			Experts – PDI vs. Cleenex		
Measure Type	**Tools**	**Measures**	**AVG**	**MEDIAN**	**P-value**	**AVG**	**MEDIAN**	**P-value**	**AVG**	**MEDIAN**	**P-value**
Debugger Usage	Cleenex	# Debugger uses in Task1	1.58	1.50	6.93E-02	1.67	1.50	**5.21E-03**	2.75	2.00	**1.77E-05**
	Cleenex	# Debugger uses in Task2	0.00	0.00	**1.00E-05**	0.00	0.00	**1.00E-05**	0.00	0.00	**1.00E-05**
User Perception	Cleenex	Answers to PS2	5.67	6.00	**2.28E-02**	6.33	6.50	**5.21E-03**	6.63	7.00	**4.79E-04**
	Other Tool	Answers to PS2	3.25	3.00	4.95E-01	4.58	5.50	5.88E-02	5.25	6.00	7.45E-02
	Tools Difference	Answers to PS2	2.42	2.00	**9.77E-04**	1.75	1.50	**4.69E-02**	1.38	1.00	6.25E-02
	Overall	Answers to UPF3	6.08	6.00	**2.75E-02**	6.58	7.00	**4.82E-04**	1.38	1.00	**4.45E-04**

Table 11. Metrics and Answers Related to Overall Analysis and User Preferences, P-values <= 0.05, which Pass the Statistical Test, Are Bolded

			Cleenex vs. OpenRefine			Cleenex vs. PDI			Experts – PDI vs. Cleenex		
Measure Type	**Tools**	**Measures**	**AVG**	**MEDIAN**	**P-value**	**AVG**	**MEDIAN**	**P-value**	**AVG**	**MEDIAN**	**P-value**
TAM	Cleenex	Sum TAM Usefulness Answers	34.67	36.00	6.02E-01	39.67	40.00	**1.37E-02**	33.00	32.00	N/A
	Other Tool	Sum TAM Usefulness Answers	23.00	23.00	5.19E-01	29.67	32.00	3.00E-01	N/A	N/A	N/A
	Tools Difference	Sum TAM Usefulness Answers	11.67	11.00	**3.91E-03**	10.00	8.00	**5.86E-03**	N/A	N/A	N/A
	Cleenex	Sum TAM Ease of Use Answers	35.08	36.00	2.06E-01	37.08	38.00	6.14E-01	31.50	33.50	N/A
	Other Tool	Sum TAM Ease of Use Answers	23.33	23.00	8.84E-02	28.75	26.00	3.35E-01	N/A	N/A	N/A
	Tools Difference	Sum TAM Ease of Use	11.75	10.50	**9.77E-04**	8.33	6.00	**3.91E-03**	N/A	N/A	N/A
User Preference	Overall	Answers to UPF1	6.17	6.00	**1.53E-02**	5.75	6.00	**3.80E-02**	4.13	4.00	**4.37E-02**
		Answers to UPF3	6.08	6.00	**2.75E-02**	6.58	7.00	**4.82E-04**	6.00	6.50	**4.45E-04**
		Answers to UPF5	5.75	6.00	**1.72E-02**	6.42	7.00	**1.64E-03**	5.25	6.50	**1.07E-02**

about user tool preference are greater for Cleenex than for OpenRefine and PDI, and, when applicable, are statistically significant. Those answers indicate that users prefer to perform data cleaning tasks with Cleenex than with any of the other two tools.

— For the open answers (UPF2, UPF4, UPF6, and UPF7 listed in Table 5), the users highlight Cleenex components that help in designing the data cleaning program and manual cleaning data, such as the graph and code editors, the debugger, and the **Manual Data Repairs (MDRs)** that do not exist in OpenRefine and PDI.

6 DISCUSSION AND LIMITATIONS

In this section, we discuss the results obtained in Section 5 and describe the limitations of the two experimental studies, one with simulated and another with real users.

User Feedback Support in Cleenex Using a Simulated User. Based on the experiments, we concluded that the Cleenex components that support user feedback reduce user effort. The components' impact depends on the data and programs in which they are used. The fact that, when using **Quality Constraints (QCs)** and MDRs, a user can repair data at any stage of a data cleaning program (represented as a graph in Cleenex) prevents quality problems from propagating through the program. Another noteworthy effect is that the type of repair has shifted from inserting whole tuples to tuple updates and deletes, which leads to less keyboard typing. As a consequence, based on HCI literature (e.g., Reference [48]), it reduces errors. Rules supported in QCs enable the reduction of tuples that users have to visualize. During an iterative execution of a data cleaning program, the MDRs persistence component prevents that all repairs need to be reapplied to the program, and the MDRs conflict resolution is able to recover repairs involved in a conflict using our proposed algorithm. Additionally, in real settings with humans, we expect that these components lead to the prevention of errors, as users suffer from less fatigue and repetitive tasks.

User Involvement Support in a Data Cleaning Process with Real Users. Cleenex programs are easier to understand than OpenRefine and PDI programs. We believe this happens because

Cleenex offers more concise programs (uses fewer operators for the same program) [36], the code editor is integrated into the main GUI screen as highlighted by the participants, and the debugger is used consistently.

As explained above, the Cleenex components that support QCs/MDRs lead to less user effort and fewer visualizations when compared to other tools, which explains the differences in task completion times and measured clicks. The differences in the number of keys typed do not hold statistical significances, but were positive.

We made sure that the data and programs used could be 100% manually cleaned using each of the tools. As a result, most participants completed Task 2 using any tool. There was no statistically significant difference in Recall between Cleenex and OpenRefine, and no notable quality measure was identified compared to PDI. Unexpectedly, participants did not use the debugger to help them manually clean data; this can be explained by the type of data quality problem (i.e., approximate duplicates), which could be cleaned with information in the current table and did not require further explorations like checking the source of some dirty tuple. However, we expect that if the table containing dirty tuples lacks sufficient information to manually clean them, then the users will utilize the debugger similarly as they have used during the experiments to understand the program.

Limitations. The number of samples directly impacts statistical significance. Although our participant count (i.e., samples) was the highest to this moment for this type of experiment in data cleaning, having an even higher number of participants may lead to statistical significance in metrics that report positive values but lack significance in our experiments, such as the number of keys pressed for Tasks 1 and 2, and the output quality measures (Precision, Recall, F1-measure) for Task 2.

Although we used the most typical data cleaning transformations to compose the programs, the advantages of Cleenex over other tools may vary, depending on the operations required and extent of the task, both of which are influenced by the quality problems present in the data. Based on our experimental results, it is not possible to compare PDI vs. OpenRefine. The reason is that the users experimenting with PDI used a slightly smaller data cleaning program, because PDI does not support approximate duplicate detection natively, i.e., without programming a new operator. To account for this difference, programs in Cleenex were reduced for the experiments involving PDI.

While the method employed (i.e., half of the users initiate with a different tool) minimizes the learning effect, this effect still could be present. Some individuals could benefit more from this effect than others in some tasks. Despite this, we tried to balance participants across the two groups and conducted thorough statistical tests.

The participants watched a short tutorial (10–20 mins) for each tool. The benefits of using a tool over another in the long term and the learning curves of the tools could not be accessed.

Our study focused on the two main types of tools used for data cleaning: ETL and data wrangling tools. For each, an open source tool was selected for evaluation. The study is limited in the number of tools used, and some fluctuations in user performance are expected within each tool type.

This study may have a limited shelf life. The findings should be interpreted in the context of the technological landscape at the time of the study and may not fully capture the advancements in software tools used for data cleaning that could occur in the future.

Scalability and performance were out of the scope of this study and were not evaluated. Delays in data processing will have a major impact on which tool the users will prefer. However, the data transformations used are similar among tools, and similar performances can be worked out if needed, as the implementations are available and open source.

The experiments were conducted in Portugal with predominantly Portuguese participants. However, we anticipate that there should be no significant difference in the performance of individuals from other nationalities, as any distinctions are more likely to be related to background factors.

7 CONCLUSIONS AND FUTURE WORK

To support the user involvement during the execution of a data cleaning program, we performed the following main contributions:

— Cleenex, a data cleaning framework that extends Ajax by incorporating **Quality Constraints (QCs)** and **Manual Data Repairs (MDRs)**, enabling users to manually edit data.
— An algorithm for conflict detection and automatic recovery of MDR instances and corresponding software components that enable the iterative execution of data cleaning programs with automatic re-application of manual data repairs.
— An extensive user involvement evaluation that includes two studies: (i) the evaluation of the Cleenex components that support user feedback in a data cleaning process with a simulated user; and (ii) the evaluation of the user involvement support in a data cleaning process for Cleenex, OpenRefine, and **Pentaho Data Integration (PDI)**, with real users.

When evaluating the Cleenex components that support the incorporation of user feedback using a simulated user, we verified that performing a data cleaning process with the full set of components (QCs, MDRs, MDRs persistence, and MDRs recovery) reduces the user effort when cleaning data.

When evaluating Cleenex against the other two tools (OpenRefine and PDI) regarding user involvement, we made the following main conclusions:

— Understanding data cleaning programs is easier if specified in Cleenex.
— The use of Cleenex leads to less time and effort to obtain clean data.
— Users have a higher preference to use Cleenex to perform a data cleaning task.
— Users highlight the main innovations of Cleenex such as the MDRs, and the Debugger that the other tools used in the study do not support.

To further improve the support of user feedback in Cleenex, the next big research task would be to introduce Machine Learning techniques that identify and automate cleaning patterns that the user can provide during the execution of the data cleaning process. Such techniques would replace the user feedback when manually editing large quantities of data and conduct a still-effective data cleaning process. Furthermore, we consider it important to further research the efficient support of concurrent user feedback, when multiple experts clean data during the same data cleaning process.

ACKNOWLEDGMENTS

The authors would like to thank the following individuals for performing their Msc Thesis on the CLEENEX prototype leading to important improvements and new features: José Esteves, Telma Fernandes, Tiago Francisco, and Cátia Ormonde. Finally, we would like to thank the reviewers for several excellent suggestions.

REFERENCES

[1] Mohamed Abdelaal, Rashmi Koparde, and Harald Schoening. 2023. AutoCure: Automated tabular data curation technique for ML pipelines. In *aiDM@SIGMOD*.
[2] Ahmad Assadi, Tova Milo, and Slava Novgorodov. 2018. Cleaning data with constraints and experts. In *WebDB@SIGMOD*.
[3] José Barateiro and Helena Galhardas. 2005. A survey of data quality tools. *Daten.-Spektr.* 14, 15-21 (2005), 48.
[4] Catriel Beeri and Moshe Y. Vardi. 1984. A proof procedure for data dependencies. *J. ACM* 31, 4 (1984), 718–741.
[5] Leopoldo Bertossi. 2019. Database repairs and consistent query answering: Origins and further developments. In *PODS*.
[6] Philip Bohannon, Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. 2007. Conditional functional dependencies for data cleaning. In *ICDE*.
[7] Loreto Bravo, Wenfei Fan, and Shuai Ma. 2007. Extending dependencies with conditions. In *VLDB* (2007), 243–254.
[8] Fei Chiang and Renee J. Miller. 2011. A unified model for data and constraint repair. In *ICDE*.

[9] Peter Christen. 2008. FEBRL—A freely available record linkage system with a graphical user interface. In *HDKM*.

[10] Gao Cong, Wenfei Fan, Floris Geerts, Xibei Jia, and Shuai Ma. 2007. Improving data quality: Consistency and accuracy. In *VLDB* (2007), 315–326.

[11] Michele Dallachiesa, Amr Ebaid, Ahmed Eldawy, Ahmed Elmagarmid, Ihab F. Ilyas, Mourad Ouzzani, and Nan Tang. 2013. NADEEF: A commodity data cleaning system. In *SIGMOD*.

[12] Fred D. Davis. 1989. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quart.* 13, 3 (1989), 319.

[13] Ronald Fagin. 1977. Functional dependencies in a relational database and propositional logic. *IBM J. Res. Devel.* 21, 6 (1977), 534–544.

[14] Wenfei Fan, Hong Gao, Xibei Jia, Jianzhong Li, and Shuai Ma. 2011. Dynamic constraints for record matching. *VLDB J.* 20, 4 (2011), 495–520.

[15] Wenfei Fan and Floris Geerts. 2012. *Foundations of Data Quality Management.* Vol. 4. Morgan & Claypool. 1–217 pages.

[16] Helena Galhardas, Daniela Florescu, Dennis Shasha, Eric Simon, and Cristian-Augustin Saita. 2001. Declarative data cleaning: Language, model, and algorithms. In *VLDB*.

[17] Helena Galhardas, Antónia Lopes, and Emanuel Santos. 2011. Support for user involvement in data cleaning. In *DaWak*.

[18] Floris Geerts, Giansalvatore Mecca, Paolo Papotti, and Donatello Santoro. 2013. The LLUNATIC data-cleaning framework. *PVLDB* 6, 9 (2013), 625–636.

[19] Stella Giannakopoulou, Manos Karpathiotakis, and Anastasia Ailamaki. 2020. Cleaning denial constraint violations through relaxation. In *SIGMOD*.

[20] Jian He, Enzo Veltri, Donatello Santoro, Guoliang Li, Giansalvatore Mecca, Paolo Papotti, and Nan Tang. 2016. Interactive and deterministic data cleaning. In *SIGMOD*.

[21] Yu Huang, Mostafa Milani, and Fei Chiang. 2020. Privacy-aware data cleaning-as-a-service. *Inf. Syst.* 94 (2020).

[22] Sean Kandel, Andreas Paepcke, Joseph M. Hellerstein, and Jeffrey Heer. 2011. Wrangler: Interactive visual specification of data transformation scripts. In *CHI*.

[23] Bojan Karlaš, Peng Li, Renzhi Wu, Nezihe Merve Gürel, Xu Chu, Wentao Wu, and Ce Zhang. 2020. Nearest neighbor classifiers over incomplete information: From certain answers to certain predictions. *PVLDB* 14, 3 (2020), 255–267.

[24] Sanjay Krishnan, Daniel Haas, Michael J. Franklin, and Eugene Wu. 2016. Towards reliable interactive data cleaning: A user survey and recommendations. In *HILDA@SIGMOD*.

[25] Sanjay Krishnan, Jiannan Wang, Eugene Wu, Michael J. Franklin, and Ken Goldberg. 2016. ActiveClean: Interactive data cleaning for statistical modeling. *PVLDB* 9, 12 (2016), 948–959.

[26] Jonathan Lazar, Jinjuan Heidi Feng, and Harry Hochheiser. 2017. *Research Methods in Human-Computer Interaction* (2nd ed.). Morgan Kaufmann.

[27] Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics Doklady*, Vol. 10. 707–710.

[28] Mohammad Mahdavi and Ziawasch Abedjan. 2020. Baran: Effective error correction via a unified context representation and transfer learning. *PVLDB* 13, 12 (2020), 1948–1961.

[29] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. Byers. 2011. *Big Data: The Next Frontier for Innovation, Competition, and Productivity.* Technical Report. McKinsey Global Institute. 156 pages.

[30] Mashaal Musleh, Mourad Ouzzani, Nan Tang, and AnHai Doan. 2020. CoClean: Collaborative data cleaning. In *SIGMOD*.

[31] Avanika Narayan, Ines Chami, Laurel Orr, and Christopher Ré. 2022. Can foundation models wrangle your data? *PVLDB* 16, 4 (2022), 738–746.

[32] Paulo Oliveira, Fátima Rodrigues, and Helena Galhardas. 2005. A taxonomy of data quality problems. In *Workshop on Data and Information Quality*.

[33] Cátia Borges Ormonde. 2017. *CLEENEX: Iterative Data Cleaning with User Intervention.* Master's Thesis. Instituto Superior Técnico, Universidade de Lisboa.

[34] Jinfeng Peng, Derong Shen, Nan Tang, Tieying Liu, Yue Kou, Tiezheng Nie, Hang Cui, and Ge Yu. 2022. Self-supervised and interpretable data cleaning with sequence generative adversarial networks. *PVLDB* 16, 3 (2022), 433–446.

[35] João L. M. Pereira. 2023. *Towards Effective and Effortless Data Cleaning: From Automatic Approaches to User Involvement.* Ph. D. Dissertation. Instituto Superior Técnico, Universidade de Lisboa.

[36] João L. M. Pereira and Helena Galhardas. 2017. Approximate duplicate elimination using state-of-the-art tools: A comparison. In *INFORUM*.

[37] Aurélien Personnaz, Sihem Amer-Yahia, Laure Berti-Equille, Maximilian Fabricius, and Srividya Subramanian. 2021. DORA the explorer: Exploring very large data with interactive deep reinforcement learning. In *CIKM*.

[38] Minh Pham, Craig A. Knoblock, and Jay Pujara. 2019. Learning data transformations with minimal user effort. In *IEEE Big Data*.

[39] Erhard Rahm and Hong Hai Do. 2000. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.* 23, 4 (2000), 3–13.

[40] Protiva Rahman, Courtney Hebert, and Arnab Nandi. 2018. ICARUS: Minimizing human effort in iterative data completion. *PVLDB* 11, 13 (2018), 2263–2276.

[41] Vijayshankar Raman and Joseph M. Hellerstein. 2001. Potter's wheel: An interactive data cleaning system. In *VLDB*.

[42] Timo Räth, Ngozichukwuka Onah, and Kai-Uwe Sattler. 2023. Interactive data cleaning for real-time streaming applications. In *HILDA@SIGMOD*.

[43] Theodoros Rekatsinas, Xu Chu, Ihab F. Ilyas, and Christopher Ré. 2017. HoloClean: Holistic data repairs with probabilistic inference. *PVLDB* 10, 11 (2017), 1190–1201.

[44] El Kindi Rezig, Mourad Ouzzani, Walid G. Aref, Ahmed K. Elmagarmid, Ahmed R. Mahmood, and Michael Stonebraker. 2021. Horizon: Scalable dependency-driven data cleaning. *PVLDB* 14, 11 (2021), 2546–2554.

[45] El Kindi Rezig, Mourad Ouzzani, Ahmed K. Elmagarmid, Walid G. Aref, and Michael Stonebraker. 2019. Towards an end-to-end human-centric data cleaning framework. In *HILDA@SIGMOD*.

[46] Barna Saha and Divesh Srivastava. 2014. Data quality: The other face of big data. In *ICDE*.

[47] Babak Salimi, Luke Rodriguez, Bill Howe, and Dan Suciu. 2019. Interventional fairness: Causal database repair for algorithmic fairness. In *SIGMOD*.

[48] Helen Sharp, Yvonne Rogers, and Jenny Preece. 2007. *Interaction Design: Beyond Human Computer Interaction* (2nd ed.). John Wiley & Sons, Inc., Hoboken, NJ.

[49] Susan Moore. 2018. How to create a business case for data quality improvement. Gartner. https://www.gartner.com/smarterwithgartner/how-to-create-a-business-case-for-data-quality-improvement

[50] Nikki Swartz. 2007. Gartner warns firms of "dirty data." *Inf. Manag. J.* 41, 3 (2007), 6.

[51] Saravanan Thirumuruganathan, Laure Berti-Equille, Mourad Ouzzani, Jorge-Arnulfo Quiane-Ruiz, and Nan Tang. 2017. UGuide: User-guided discovery of FD-detectable errors. In *SIGMOD*.

[52] Maksims Volkovs, Fei Chiang, Jaroslaw Szlichta, and Renée J. Miller. 2014. Continuous data cleaning. In *ICDE*.

[53] David Vos, Till Döhmen, and Sebastian Schelter. 2022. Towards parameter-efficient automation of data wrangling tasks with prefix-tuning. In *TRL@NeurIPS*.

[54] Xi Wang and Chen Wang. 2020. Time series data cleaning: A survey. *IEEE Access* 8 (2020), 1866–1881.

[55] Mohamed Yakout, Ahmed K. Elmagarmid, Jennifer Neville, Mourad Ouzzani, and Ihab F. Ilyas. 2011. Guided data repair. *PVLDB* 4, 5 (2011), 279–289.

[56] Zhuoran Yu and Xu Chu. 2019. PIClean: A probabilistic and interactive data cleaning system. In *SIGMOD*.

Received 14 March 2023; revised 23 January 2024; accepted 25 January 2024