Effective Data Cleaning with Continuous Evaluation

Ihab F. Ilyas
University of Waterloo
ilyas@uwaterloo.ca

Abstract

Enterprises have been acquiring large amounts of data from a variety of sources to build their own "Data Lakes", with the goal of enriching their data asset and enabling richer and more informed analytics. The pace of the acquisition and the variety of the data sources make it impossible to clean this data as it arrives. This new reality has made data cleaning a continuous process and a part of day-to-day data processing activities. The large body of data cleaning algorithms and techniques is strong evidence of how complex the problem is, yet, it has had little success in being adopted in real-world data cleaning applications. In this article we examine how the community has been evaluating the effectiveness of data cleaning algorithms, and if current data cleaning proposals are solving the right problems to enable the development of deployable and effective solutions.

1 Introduction

Data collection and acquisition often introduce errors in data, for example, missing values, typos, mixed formats, replicated entries of the same real-world entity, and even violations of business rules. Figure 1 shows examples of data errors in a relational table, where multiple types of errors co-exist in the same data set. Even when data is machine-generated (e.g., data center's logs and sensor readings) errors still occur due to device malfunctions, interrupted connections, or due to aggregating data from various data sources.

With the recent surge in the availability of a variety of big data technologies and tools that allow fast ingestion of a large amount of data in various formats, dirty data has become the norm rather than the exception. Not surprisingly, developing data quality solutions is a challenging topic and is rich with deep theoretical and engineering problems. A large body of work has addressed some of the well-defined problems; for example, refer to the survey [11] on cleaning relational data with violations of integrity constraints, and to the multiple surveys on record linkage and de-duplication [14, 9, 8, 15, 10], and also on data profiling [1]. Most of this research focuses primarily on algorithms and techniques to detect certain types of errors, and a smaller subset of the solutions provide automatic or semi-automatic techniques to clean the detected errors.

However, most of the available solutions suffer from fundamental pragmatic problems when applied to real-world enterprise dirty data. To list a few, (1) *scale* renders algorithms with quadratic complexity (or worse) infeasible in big data settings—for example, those algorithms that enumerate all pairs of records to assess if a violation of a functional dependency exists among pairs of records, or to detect duplicate pairs; (2) *involving humans* in the data cleaning cycle is often mandated by business owners and is necessary when dealing with

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

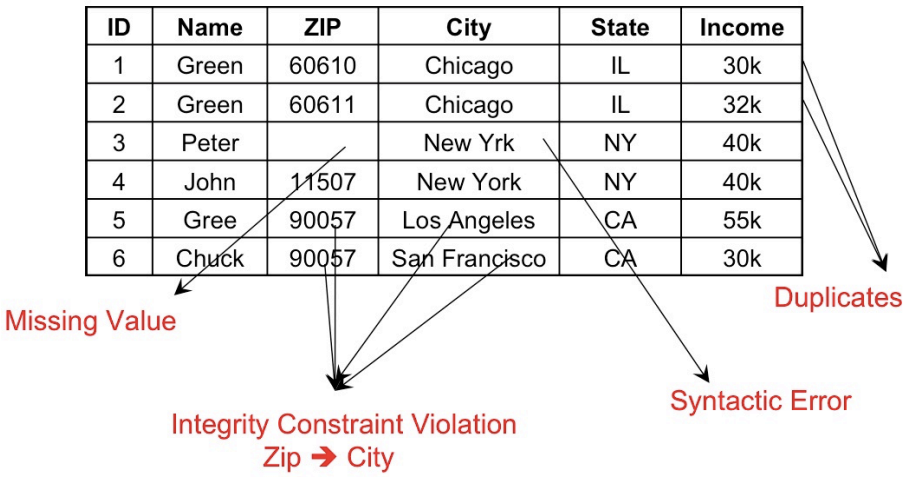

Figure 1: Example Data Errors

critical data, where ad-hoc automatic cleaning criteria are neither trusted nor accepted (we elaborate more on this in the rest of the paper); (3) *data variety* is often as serious as data volume in today's "data lakes": dealing with multiple formats, lack of unified schema, and the need for a rich set of transformations becomes crucial to carry out the cleaning process; (4) *holistic cleaning* of the data with respect to multiple quality problems is an effective way to spot the erroneous portion of the data; trying to clean each of the problems in isolation often results in wrong guesses of where errors are due to the limited context of these algorithms; and (5) *expressing business rules and sanity checks* is not often performed in terms of data quality and integrity constraints on the data sources, rather, these rules are usually expressed on top of aggregates and analytics (e.g., SQL views) computed from the raw source data; this decoupling often makes it very difficult to assess the effectiveness of cleaning tools and their impact on the business value of this data.

The scale challenge is often addressed using a variety of techniques including the use of sampling [17], approximate cleaning by partitioning the data into blocks as often done in data deduplication [3], and by adopting scale-out computing infrastructure such as Spark [19]. On the other hand, the variety in data syntax is often tackled by developing a large number of "connectors" that allow for accessing many external data sources with proprietary formats. It is less obvious how the rest of the challenges will be handled when developing an industry-strength data cleaning solutions.

In this paper, we will focus on a couple of these challenges, mainly, the problem of the decoupling in space and time between detecting data errors and the repairing of these errors, which is also tied to the problem of involving humans in guiding and evaluating data cleaning solutions. Furthermore, we argue in Section 3 that a continuous data cleaning life-cycle, with humans in the loop, is necessary to address the decoupling of error detection and error correction.

The rest of this article is organized as follows: in Section 2, we present some of the commonly used cleaning objective functions or evaluation criteria in most of the current solutions, and we discuss the challenges with these objectives. In Section 3, we give an example of a pragmatic solution to convey the impact of effectively involving data analysts and business users in guiding and evaluating the cleaning solution. In Section 4, we describe a real world deployment of a commercial data curation solution, and we conclude in Section 5 with a summary and final remarks.

2 Data Cleaning Objective Functions

Data cleaning primarily involves primarily two main exercises: (1) *Error Detection*, where errors (such as duplicates or violations of business rules) are spotted and surfaced to be examined either by a human or by a repairing mechanism; and (2) *Data Repairing*, which involves updating the data sources to remove these errors, for example, deleting erroneous records, merging duplicate records, or updating specific values that are believed to cause the violations.

Error detection can be performed in a variety of ways, ranging from "eye-balling" the data by experts to automatic tools and algorithms to highlight possible errors in the data with respect to some objective function. For example, to automatically detect errors with respect to declared integrity constraints (such as functional dependencies or denial constraints [6]), error detection algorithms often enumerate possible combinations of records to create context for checking a rule: to detect possible duplicates, all pairs of records might need to be generated to apply a classifier or a similarity function to detect duplicates. For other types of errors, for example, outliers, statistical properties of the underlying data are often calculated to decide on "unusual" values.

Unfortunately, *data repairing* is often more challenging; asking humans and experts to correct all possible errors spotted by the error detection tools is obviously prohibitively expensive in large data settings. On the other hand, master or reference data sets that can be used to guide the repairing process might not be available and often suffer from poor coverage of the data [7]. Much work adopt other guidelines to suggest data repairs based on less-justified principles such as *minimality of repairs* [5, 13]. The general intuition is that since there are more clean data than errors, then if algorithm A changes more data than Algorithm B to remove data errors (with respect to some chosen error detection algorithm), Algorithm B is more preferred. While the concept makes sense, it is not obvious if it can be translated into an operational objective function, where the minimum number of cell updates to a dirty database produces "the clean" version of this data.

In previous work [2], we conducted an experiment to measure the precision and the recall of multiple clean instances produced by an automatic repairing algorithm. All these instances enjoy some notion of minimality of updates (in this case cardinality-set minimal). Figure 2 plots the precision and recall for each of these minimal repairs. Besides the low values for precision and recall, the figure further shows a spread in the values of precision and recall of these repairs. These results suggest that there is little to no correlation between minimality-based repairing and accuracy of the repairs. We argue that the reason for these results is the way we used minimality as operational semantics of data repairing as opposed to a guiding principle. In other words, believing that "there is more good than bad in the data" does not directly translate into favouring cleaning operations with fewer changed cells, especially when performed in local contexts and without fully understanding the sources of theses errors or how these errors interact. We show in Section 3 that minimality can be a very useful concept in ranking and evaluating the likelihood of errors as long as humans have the final decision in the repairing process.

From these observations, and since data cleaning is a continuous process, evaluating the effectiveness of data cleaning activity is essential in guiding error detection and repairing of erroneous data. When a "ground truth" is available or can be obtained, the accuracy of the data cleaning (detection and repairing) can be measured via standard measures such as precision and recall. However, in the absence of such ground truth, it is not always obvious how to measure the effectiveness of the proposed cleaning technique. We briefly discuss two main approaches:

- *Sample and verify by experts:* to address the scale problem of involving humans in the verification of data repairs, one can present a sample of the repairs (data updates) to be examined by human experts. Several challenges need to be addressed when using samples for verification, for example, (1) how to provide enough context to humans to judge the correctness of the repairs; while this might not be crucial for certain types of errors such as the check constraint $Salary > 45k$, other errors like outliers require more "global" context to surface for the expert; (2) the normal challenges involved in using human experts

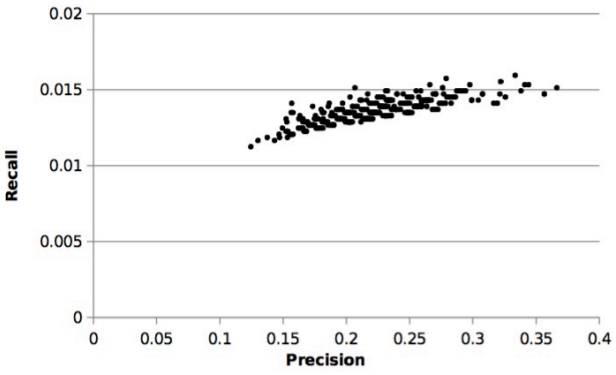

Figure 2: Accuracy of Minimal Repairs

such as expertise registration, handling conflicting opinions, and dealing with access controls; and (3) how to extrapolate and scale the effectiveness measure from measures calculated from the sample, which can vary in complexity depending on the type of errors addressed.

- *Assessing the impact on reports and analytics:* Often, the raw data, where data is generated or acquired, might not have enough semantics in the form of declared integrity constraints to even detect errors, and errors can only be spotted by analysts and business users at the reporting or analytics layer. This complicates both error detection and evaluating the impact of the repairing strategy employed. Several challenges arise, but mainly (1) how to push spotted errors in reports down to where errors can be corrected and repaired? and (2) how to effectively propagate the effect of the repairs up to the reporting layer to guide and to evaluate the impact of the repairing process? are of utmost importance to address.

While discussing human involvement and sampling for data cleaning is outside the scope of this article, in Section 3, we focus on pragmatic approaches to assess the impact of the data cleaning tools on reports and analytics.

3 Measuring The Effectiveness of Cleaning Solutions

Through our experience with deploying Tamr [1] (the commercial descendant of the Data Tamer System [16]) in multiple large enterprises (e.g., an international automotive manufacturer) with large-scale data cleaning projects, it was evident that one of the biggest challenges in adopting a data cleaning approach is measuring the impact of this solution on the main line of business (aka Return on Investment), which decides (1) the amount of resources to be dedicated to the cleaning project, (2) the incentives to involve domain experts in the curation cycle, and (3) the prioritization of the type of anomalies to fix. For the rest of this section, we assume data errors that can be detected by either human involvement, or by automatic techniques that spot conflicts with other database values (e.g., as in the case of duplicates and violations of integrity constraints).

To address the challenge of assessing the effectiveness of a proposed data cleaning approach, we focus on the decoupling in space (the data set) and time (the data processing phase) between error detection and data repairing efforts [4]: most errors are born during data generation or data acquisition, due to a variety of error generation models (e.g., human errors, using the wrong transformations, or as a result of integrating data from multiple data sources). However, data is almost never consumed in its raw format; instead, layers of transformations, aggregation, and analytics are often in place before data is consumed by business analysts or data scientists

[1] www.tamr.com

in the form of reports, or in rich visualizations layers. Business analysts and data scientists can easily spot inconsistencies, fishy results, violations of business rules, and various types of errors. In other words, while detection makes sense much later in data processing pipelines on views built on top of the source data (with the availability of rich semantics and domain experts), errors need to be corrected much earlier in the data processing life cycle, directly on the raw data sources.

Example 1: To illustrate the problem, we use a simple example, where data sources are relational tables and the "analytics" is a simple aggregation query to compute a target view on this data: consider the report T in Figure 3 about shops for an international franchise. T is computed according to the following query on two sources $Shops$ and $Emps$:

Q:
```
SELECT SId as Shop, Size, Grd, AVG(Sal) as
  AvgSal, COUNT(EId) as #Emps,'US' as Region
  FROM US.Emps JOIN US.Shops ON SId
  GROUP BY SId, Size, Grd
```

The HR business analysts enforce a set of policies in the franchise workforce and identify two problems in T. The first violation (t_a and t_b, in bold) comes from the rule "in the same shop, the average salary of the managers (GRD=2) should be higher than that of the staff (GRD=1)". The second violation (t_b and t_d, in italic) comes from the rule " a bigger shop cannot have a smaller number of staff". Note that no business rules are defined on the sources $Shops$ and $Emps$.

T	Shop	Size	Grd	AvgSal	#Emps	Region
t_a	**NY1**	46 ft^2	**2**	**99** $	1	US
t_b	**NY1**	*46* ft^2	***1***	**100** $	*3*	US
t_c	NY2	62 ft^2	2	96 $	2	US
t_d	NY2	*62* ft^2	*1*	90 $	*2*	US
t_e	LA1	35 ft^2	2	105 $	2	US

Emps	EId	Name	Dept	Sal	Grd	SId	JoinYr
t_1	e4	John	S	91	1	NY1	2012
t_2	e5	Anne	D	99	2	NY1	2012
t_3	e7	Mark	S	93	1	NY1	2012
t_4	e8	Claire	S	116	1	NY1	2012
t_5	e11	Ian	R	89	1	NY2	2012
t_6	e13	Laure	R	94	2	NY2	2012
t_7	e14	Mary	E	91	1	NY2	2012
t_8	e18	Bill	D	98	2	NY2	2012
t_9	e14	Mike	R	94	2	LA1	2011
t_{10}	e18	Claire	E	116	2	LA1	2011

Shops	SId	City	State	Size	Started
t_{11}	NY1	New York	NY	46	2011
t_{12}	NY2	New York	NY	62	2012
t_{13}	LA1	Los Angeles	CA	35	2011

Figure 3: A report T on data sources Emps & Shops.

Example 1 clearly highlights the following:

1. Errors can be easily spotted when richer semantics are available to detect violations of integrity constraints, which can be done either automatically or by involving human experts who are familiar with a given report.

2. Data repairing cannot be done at the report level, since these are usually read-only views that can be arbitrarily complex. Data curation tools and solutions will only be applicable to raw data sources.

3. Repairing algorithms on the data sources will be evaluated based on their ability to correct the errors spotted in the computed reports and not solely on the number of cells they change in the source data. Note, however, that minimality of repairs can be used as a guiding principle and a ranking mechanism of more probable repairs to surface for experts.

We argue that any effective cleaning solution has to address two main challenges: (1) closing the gap between detection and evaluation in the reporting layer, and data repairing at the raw data sources; this necessarily requires reasoning about provenance and data lineage; and (2) providing experts and evaluators with error explanations and prescriptions to clean data at the sources in a way that is relevant to and in the context of the analytics and reports of interest.

Figure 4 depicts an *evaluate-clean* loop that adopts a pragmatic cleaning evaluation (via business analysts and domain experts) and leverages provenance, automatic error detection and repairing techniques to suggest possible errors, to mine possible explanations of these errors, and to translate them into cleaning actions. The life cycle is comprised of four main phases:

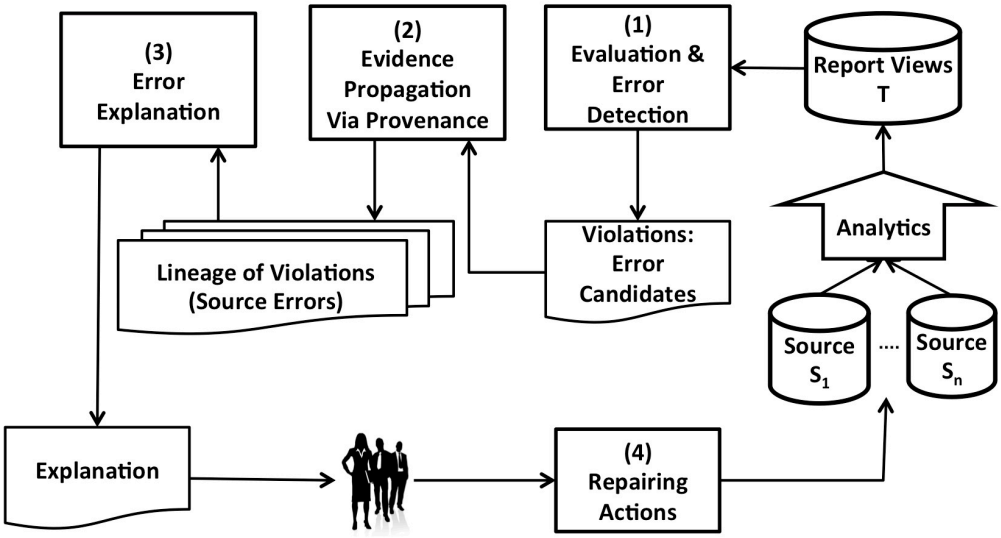

Figure 4: Clean-Evaluate Loop

1. *Evaluation and Error Detection:* Errors are detected on the reports level via normal error detection techniques such as running automatic error detection scripts, checking for violations of business rules, or visually spotting outliers [18]. This is also the phase where the reports are observed to evaluate the effect of cleaning or updating data sources (e.g., via A/B or split testing).

2. *Error Propagation:* Since violations detected in the report are actually caused by errors that crept in at an earlier stage, i.e., from the sources, propagating these errors from a higher level in the transformation to the underlying sources can help in identifying the source of the errors and in prescribing actions to correct them. Scorpion [18] and DBRx [4] are example solutions that attempt to trace back the tuples that contributed to the problems in the target. For example, Tuples t_a, t_b are in violation in T and their lineage is $\{t_1 - t_4\}$ and $\{t_{11} - t_{12}\}$ over Tables $Emps$ and $Shops$. By removing or fixing these source tuples, the violation is removed.

As the view definition gets more complex, for example, to include aggregation and user-defined functions, the provenance of the reported data gets more complex and hence, very challenging to reason about. This complicates error propagation and relating errors in data sources to errors that can be easily spotted at the analytics layer.

Error propagation is further complicated by the fact that the source values that belong to the lineage of an erroneous report value might not be all wrong, as errors may not be deterministically inferred from data lineage. For example, an aggregate query would clump together several source tuples with few containing actual errors. Source tuples do not contribute to violations in equal measure. Hence, we need a mechanism to accumulate evidence on tuples across multiple violations of business rules to identify the most likely tuples with errors. The work in [18, 4] provides mechanisms to compute the likelihood of errors in the source data, given the lineage and the type of errors detected at the reports level.

3. *Error Explanation:* After identifying the likely errors at the sources, mining concise and accurate explanations of these errors becomes an important task to allow experts to perform cleaning actions on the sources that have a direct effect on the output results.

   Error explanation is done in [4] as building a view on the data sources that covers (only) the errors propagated from reports to sources via error propagation. Among all possible views that can be built, the ones that adhere to predefined plausible properties (e.g., precision, recall, and conciseness) are chosen.

   In Example 1, two possible explanations of the problems are: $[Emps.JoinYr = 2012]$ on Table $Emps$, and $[Shops.State = NY]$ on Table $Shops$. As we can see, t_b is the only tuple that is involved in both violations and, hence, is identified by the repairing algorithm as erroneous. Its lineage is $\{t_1, t_3, t_4\}$ and $\{t_{11}\}$ over Tables $Emps$ and $Shops$, respectively. By focusing on this tuple, we can compute more precise explanations on the sources, such as $[Emps.Dept = S]$.

4. *Repairing Actions:* Based on the explanation, experts might invoke multiple actions to clean the data sources, for example, running transformation scripts for units alignment, performing targeted updates on the records identified as erroneous, or even deleting sources with low quality or redundant information. Once these actions take place, running analytics again and moving to Phase 1 of the process will allow for measuring the impact of these actions in a usage-based manner and in the context of the given analytics.

4 Real-World Experience

The aforementioned Clean-Evaluate loop in Figure 4 has been adopted by many current commercial data cleaning and curation platforms such as Tamr [2] (based on the Data Tamer system and Trifacta [3] (based on the Wrangler system [12]). For example, Tamr employs a *continuous* curation process, where user's engagement is used at multiple levels, for example: (1) to assess the quality of analytics and translate user feedback into feedback on the underlying data sources; (2) to generate training data for the back-end machine learning models, (e.g., deduplication models); and (3) to administer repairing actions on the underlying data sources given the feedback on analytics and the learned models.

Deploying this curation life-cycle in a single curation project at a Fortune 500 manufacturer involved integrating data from more than 50 ERP (Enterprise Resource Planning) systems with millions of records in a variety of formats. The curation project employed more than 20 experts for validating the detected error (e.g., duplicate records) and the suggested repairs. The result was a cleaner data set that uncovered a saving of more than $100M for the company. This exercise won't be possible without an iterative process, where error detection and repairing are driven by the analytics that can be examined by domain experts and business analysts, and

[2] www.tamr.com

[3] www.trifacta.com

anomalies are tied back to the errors in the raw data sources (in this case the ERP systems). In this example, various traditional automatic data cleaning algorithms have been re-purposed to serve as ranking mechanisms to show the most probable repairs to the human-in-the-loop, which maximize the benefit of the budget allocated to expert involvement in the project.

5 Conclusion

A key aspect of data cleaning is the ability to evaluate the accuracy and the effectiveness of the algorithms and tools used in the process. The evaluation is almost straightforward when humans examine the whole set of data cleaning results, or when ground truth can be obtained via, for example, reference data sources. However, with the large amounts of dirty data currently acquired in big data infrastructures, neither humans nor complete master data sources are available for evaluation. We highlighted multiple challenges with widely adopted cleaning objective functions such as minimality of repairs. We also showed an example workflow to make the evaluation process an integral part of the data cleaning process itself, where experts are involved in evaluating the effectiveness of the underlying cleaning tools at the report and analytics level, while automatic techniques are re-purposed to suggest ranked updates. Numerous challenges remain to be addressed in developing adoptable data cleaning platforms, including effective ways to engage users and experts in guiding and evaluating the cleaning algorithms, connecting to the continuously growing number of data sources with a variety of data formats, and developing scalable repairing algorithms that can deal with large amounts of data to suggest the most relevant repairs.

References

[1] Ziawasch Abedjan, Lukasz Golab, and Felix Naumann. Profiling relational data: a survey. *VLDB J.*, 24(4):557–581, 2015.

[2] George Beskales, Ihab F. Ilyas, and Lukasz Golab. Sampling the repairs of functional dependency violations under hard constraints. *PVLDB*, 3(1):197–207, 2010.

[3] Mikhail Bilenko, Beena Kamath, and Raymond J Mooney. Adaptive blocking: Learning to scale up record linkage. In *6th International Conference on Data Mining*, pages 87–96, 2006.

[4] Anup Chalamalla, Ihab F. Ilyas, Mourad Ouzzani, and Paolo Papotti. Descriptive and prescriptive data cleaning. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pages 445–456, 2014.

[5] Jan Chomicki and Jerzy Marcinkowski. Minimal-change integrity maintenance using tuple deletions. *Information and Computation*, 197(1/2):90–121, 2005.

[6] Xu Chu, Ihab F. Ilyas, and Paolo Papotti. Holistic data cleaning: Putting violations into context. In *29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013*, pages 458–469, 2013.

[7] Xu Chu, Mourad Ouzzani, John Morcos, Ihab F. Ilyas, Paolo Papotti, Nan Tang, and Yin Ye. KATARA: reliable data cleaning with knowledge bases and crowdsourcing. *PVLDB*, 8(12):1952–1963, 2015.

[8] Xin Luna Dong and Felix Naumann. Data fusion: resolving data conflicts for integration. *Proceedings of the VLDB Endowment*, 2(2):1654–1655, 2009.

[9] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16, 2007.

[10] Lise Getoor and Ashwin Machanavajjhala. Entity resolution: theory, practice & open challenges. *Proceedings of the VLDB Endowment*, 5(12):2018–2019, 2012.

[11] Ihab F. Ilyas and Xu Chu. Trends in cleaning relational data: Consistency and deduplication. *Foundations and Trends in Databases*, 5(4):281–393, 2015.

[12] Sean Kandel, Andreas Paepcke, Joseph M. Hellerstein, and Jeffrey Heer. Wrangler: interactive visual specification of data transformation scripts. In *Proceedings of the International Conference on Human Factors in Computing Systems, CHI 2011, Vancouver, BC, Canada, May 7-12, 2011*, pages 3363–3372, 2011.

[13] Solmaz Kolahi and Laks V. S. Lakshmanan. On approximating optimum repairs for functional dependency violations. In *ICDT*, 2009.

[14] Nick Koudas, Sunita Sarawagi, and Divesh Srivastava. Record linkage: similarity measures and algorithms. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, pages 802–803, 2006.

[15] Felix Naumann and Melanie Herschel. *An Introduction to Duplicate Detection*. Synthesis Lectures on Data Management. 2010.

[16] Michael Stonebraker, Daniel Bruckner, Ihab F. Ilyas, George Beskales, Mitch Cherniack, Stanley B. Zdonik, Alexander Pagan, and Shan Xu. Data curation at scale: The data tamer system. In *CIDR 2013, Sixth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 6-9, 2013, Online Proceedings*, 2013.

[17] Jiannan Wang, Sanjay Krishnan, Michael J Franklin, Ken Goldberg, Tim Kraska, and Tova Milo. A sample-and-clean framework for fast and accurate query processing on dirty data. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, pages 469–480, 2014.

[18] Eugene Wu and Samuel Madden. Scorpion: Explaining away outliers in aggregate queries. *PVLDB*, 6(8):553–564, 2013.

[19] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: cluster computing with working sets. In *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, volume 10, page 10, 2010.