Towards Automated Data Cleaning Workflows

Mohammad Mahdavi[1], Felix Neutatz[2],
Larysa Visengeriyeva[1], and Ziawasch Abedjan[1,2]

[1] TU Berlin, Berlin, Germany
mahdavilahijani@tu-berlin.de
larysa.visengeriyeva@campus.tu-berlin.de
abedjan@tu-berlin.de
[2] DFKI GmbH, Berlin, Germany
felix.neutatz@dfki.de

Abstract. The success of AI-based technologies depends crucially on trustful and clean data. Research in data cleaning has provided a variety of approaches to address different data quality problems. Most of them require some prior knowledge about the dataset in order to select and configure the approach correctly. We argue that for unknown datasets, it is unrealistic to know the data quality problems upfront and to formulate all necessary quality constraints in one shot. Pragmatically, the user solves data quality problems by implementing an iterative cleaning process. This incremental approach poses the challenge of identifying the right sequence of cleaning routines and their configurations. In this paper, we highlight our work in progress towards building a cleaning workflow orchestrator that learns from cleaning tasks in the past and proposes promising cleaning workflows for a new dataset. To this end, we highlight new approaches for selecting the most promising error detection routines, aggregating their outputs, and explaining the final results.

Keywords: Data Cleaning Workflows · Machine Learning · Data Profiling.

1 Data Cleaning: The Usage Gaps

Deriving value from AI- and machine learning-based technologies crucially depends on the quality of the underlying data [17]. Research in data cleaning has provided a variety of tools and approaches to address different data quality problems [6, 22, 11, 32, 18]. Nevertheless, in real-world applications, human agents utilize handcrafted scripts to curate their datasets [28, 11]. Underlying problems that impede the application of thoroughly researched cleaning algorithms are as follows:

No one-size-fits-all solution. Research on data cleaning solves well-defined data quality problems that often do not generalize to all problems of a real-world dataset. In particular, data quality problems are exposed with regard to a specific context, such as rules, dictionaries, patterns, and distributions. Current solutions only focus on only one of the contexts above [1].

Iterative data cleaning. Oftentimes, one has to perform multiple rounds of cleaning and wrangling until the data reaches a satisfactory state [27]. Moreover, some data quality problems are hidden in a way that they can only be exposed after some iterations of certain cleaning or transformation procedures. For example, missing value imputation facilitates the discovery of outliers in a dataset [8].

Trial-and-error parametrization. Current techniques require user-defined algorithm parameters, such as rules or thresholds, which are not straightforward to select by a data practitioner [26]. Often, the user has to figure these parameters out during a trial-and-error process that adds more cycles to the iterative process of data cleaning.

In this paper, we report on the conducted research and the ongoing work towards a framework that leverages machine learning and data profiling techniques to build a *cleaning workflow orchestrator* for a dataset. In particular, we are working towards a solution that

- uses similarities of current cleaning tasks with previous cleaning tasks to assess the possible gain of a certain tool on a new dataset (Section 2.1).
- enables users to aggregate the results of stand-alone cleaning strategies in a holistic manner (Section 2.2).
- featurizes data values to better explain the context of a data error and enable an active learning approach to sample more promising data values for labeling (Section 2.3).

Next, we will describe the overall architecture of our vision and shed light on some of our intermediate results, some of which have been already published [29, 15, 16].

2 Machine Learning-Driven Cleaning Pipelines

We consider a data science use case where data analytics and data preparation are carried out on a frequent basis, accumulating a history of data cleaning tasks from the past that can be logged for later analysis [13]. Furthermore, we assume that the data scientists are in possession of multiple cleaning algorithms or routines. While in our experiments, we are considering off-the-shelf data cleaning prototypes from research, any sort of custom cleaning script can be considered as an individual cleaning solution or algorithm.

Figure 1 illustrates the overall architecture of the proposed system. The first task is to identify metadata that describes the quality problems of a dataset. Thus, given a new dataset, the *Dataset Profiler* component creates a profile

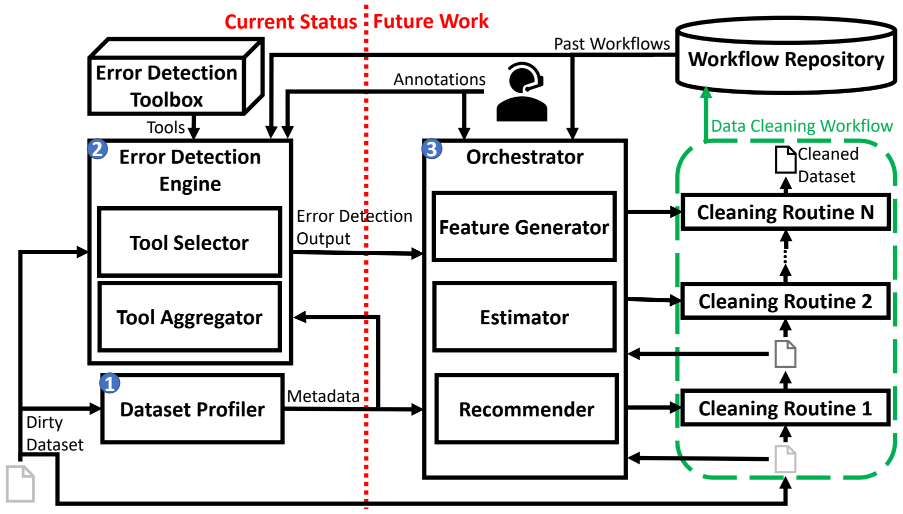

Fig. 1. Generation of data cleaning workflows includes three main steps: (1) profiling data, (2) detecting errors by identifying the most promising tools and aggregating them, and (3) generating dataset-specific cleaning workflows.

by extracting relevant metadata (Step 1). This profile summarizes the content, structure, and the dirtiness of the dataset into statistics and distributions. The *Error Detection Engine* leverages the metadata to compare the similarity of the new dataset to the previously cleaned datasets in the *Workflow Repository* (Step 2). The *Tool Selector* uses this metadata to identify the most promising error detection strategies, whose estimated performance is high enough. We will detail this step in Section 2.1. The *Error Detection Engine* then runs the promising error detection strategies on the new dataset to identify potential data quality problems. The profile of the new dataset is then enriched by adding information related to the strategies' output, such as the output size. Based on the enriched profile of the data, the set of potential cleaning algorithms can be refined. Furthermore, the *Tool Aggregator* uses the enriched dataset profile to aggregate the output of the promising error detection strategies into one final output. We will detail this step in Section 2.2. The *User* is involved in the process once the initial profiling and detection phase is over. The first task of the user is to annotate a sample of the detected errors. Leveraging a feature representation that describes each data cell, our machine learning approach propagates the user labels to other similar data values with a similar set of feature values. The generated metadata, the error detection results, and the annotations will be used by the *Orchestrator* to generate a dataset-specific cleaning workflow (Step 3). Currently, we focus on workflows as sequences of cleaning routines. More complex control flow elements, such as branches, are future work. Finally, the executed cleaning workflow can be stored in the workflow repository. In the following, we discuss insights that we have gained so far working on this project.

2.1 Configuration-Free Tool Selection

Existing data cleaning solutions are usually tailored towards one specific type of data errors, such as outliers, syntactic pattern violations, or missing values. However, cleaning the dataset might require a combination of such solutions [1]. Although the number of available data cleaning routines is limited, there is a vast space of possible configurations for each algorithm. To address this challenge, we propose an automated approach for configuring the error detection algorithms and estimating their F_1 score on a new dataset [15].

To select the proper set of cleaning routines, we use the similarity between the current task and previous data cleaning tasks. For a dataset at hand, we need to select cleaning routines that have successfully cleaned similar datasets in the past. The key challenge here is to define a similarity metric that encodes the data quality of datasets.

We have created a dirtiness profile based on data profiling features [2]. These features cover content-describing metadata, such as value distribution, and structure-describing metadata, such as character distribution [15]. We compare the similarity of datasets through these metadata to filter out irrelevant error detection algorithms and configurations that had poor accuracy on the previous similar datasets [15]. Next, we run the selected error detection routines on the new dataset to compute the second group of metadata that are based on the output of the error detection routines. The raw output of a tool on a dataset harbors relevant information, such as the output size and its overlap with the output of other tools. The dirtiness profile of the dataset will be enriched with these metadata as well. Finally, the regression models estimate the F_1 score of the selected error detection routines based on the similarity of the final dirtiness profile of the dataset to the previous datasets.

To anecdotally show that the approach is promising, we evaluate our performance estimator on 11 diverse datasets: *Hospital* [25], *Flights* [25], *Rayyan* [20], *IT* [1], *Beers* [10] are our real-world datasets that have been cleaned manually, and *Salaries* [29], *Address* [1], *Movies* [7], *Restaurants* [7], *Soccer* [3], and *Tax* [3] are our synthetic datasets. We also have 15 error detection strategies generated by configuring 7 entirely different error detection tools: NADEEF [6], OpenRefine [28], and KATARA [5] are our rule, pattern, and knowledge base violation detection systems, respectively, and Histogram, Gaussian, Gaussian Mixture, and Partitioned Histogram Modeling are our outlier detection strategies [22]. We apply the leave-one-out methodology to evaluate our approach. Each time, we consider one of the datasets as the new arriving dataset and the rest of datasets as the historical training datasets. So, our performance estimator trains regression models to learn the relationships between profile components and F_1 scores of all 15 error detection strategies and estimates the corresponding F_1 score for the new dataset. Our prototype is available online[3].

Figure 2 shows the results of our experiments. We use mean squared error (MSE) to evaluate the quality of the estimated performance of these strategies.

[3] https://github.com/bigdama/reds

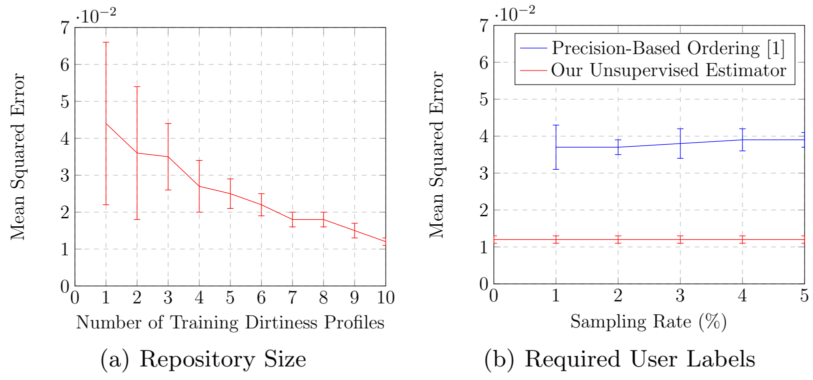

Fig. 2. MSE in estimating the F_1 score of error detection algorithms. (a) The MSE decreases with the size of the cleaning workflow repository. (b) Our unsupervised performance estimator approach predicts the F_1 score of the algorithms more accurately than the semi-supervised baseline [1].

The first experiment (a) shows how the number of existing training datasets within the repository influences the estimation accuracy of our proposed solution. Each point in the graph reports the average mean and standard deviation of 5 independent runs on estimating the performance of each of the 15 tools. As depicted, the MSE significantly decreases with the size of the workflow repository. The second experiment (b) shows that our unsupervised performance estimator approach provides more accurate estimations than the precision-based ordering approach [1] that requires additional user labels.

The described approach required manual configuration of each tool per dataset. In fact, it is possible to relieve the user also from the configuration task using our dirtiness profile-based approach. Our novel system Raha [16] first generates a range of possible configurations for each tool independent of the dataset. Based on the similarity of the new dataset to historical datasets, Raha filters out irrelevant error detection strategies for each column of the new dataset at hand.

2.2 Aggregating Error Detection Results

Although the prevalence of specific error types suggests a ranking of error detection algorithms for a dataset at hand, we cannot limit the error detection effort to only running one single approach [27]. Having access to various data cleaning solutions implicates an effective aggregation of the error detection results, which we consider as a classification problem [29]. To holistically combine error detection methods, we use state-of-the-art ensemble learning algorithms [34] while leveraging the dataset's metadata. We train an error detection classifier by creating a feature representation based on the error detection results from the different data cleaning systems and additional metadata [29].

We use the ensemble learning method *stacking* [31] to learn the error classification model. Stacking is an approach for training a *meta-learner* by combining

	Algorithms	P	R	F_1
Individual Cleaning Strategies	Wrangler [11]	0.41	0.14	0.21
	Nadeef (Dedup) [6]	0.32	0.11	0.17
	Outlier detection (Histograms) [22]	0.28	0.01	0.02
	Outlier detection (Gaussian) [22]	0.32	0.18	0.23
	Nadeef (FD) [6]	0.52	0.18	**0.27**
Aggregators	Majority wins	0.59	0.02	0.03
	Union all	0.37	0.47	**0.42**
	Precision-based ordering (with δ=0.1) [1]	0.36	0.46	0.40
Our Approach	**Stacking with metadata**	**0.56**	**0.91**	**0.69**

Table 1. Precision (P), Recall (R), and F_1 scores of individual error detection algorithms and aggregators on the *Address* dataset. The Stacking approach exposes the benefits of accumulating metadata into the aggregation of individual cleaning strategies: our approach achieves 0.91 Recall as a result of augmenting the systems results with the datasets metadata.

multiple *first-level* models for error detection. The main idea is to train the first-level learners on the same training dataset, and then generate a new training dataset for the meta-learner. The outputs of the first-level learners constitute the input for training the meta-learning model. In our initial prototype, we train three different first-level classifiers on the same feature vector: a *neural network* with one hidden layer, a *decision tree,* and a *naive Bayes* classifier. Each of these models classifies dataset cells as *erroneous* or *correct.* The meta-classifier, *logistic regression*, is trained then on the produced output of the first-level classifiers.

Table 1 shows the performance scores of our ensemble learning aggregation based on *Stacking* in comparison to individual error detection strategies from the literature as well as standard aggregators, such as *Majority Wins, Union All*, and *Precision Based Ordering* [1]. The experiments were performed on the above-mentioned *Address* dataset, which contains misspelled or missing values, and field separation violations, resulting in 36% of errors in the whole dataset. Our prototype is available online[4].

As metadata provides signals on individual values and inter-column relationships, they support the four major types of error detection strategies: *pattern violation detection*, *rule violation detection*, *outlier detection*, and *duplicate-based error detection* [1]. To improve the performance of error classification, we incorporate metadata, such as *data type affiliation, attribute domain, frequency, null values*, and *multi-column dependencies*, into our learning algorithms. Initially, the classifiers are trained on the feature vectors that comprise the output of individual error detection algorithms and the metadata information. Operating on the augmented feature vector explains why the stacking learning approach results in a higher recall, i.e., 0.91%, compared to the sum of the recalls of the individual error detection methods. Using the metadata, the classifier leverages more information to classify the dataset cells. This experiment demonstrates that

[4] http://bit.ly/systems-aggregation

Source	Flight	Departure	Arrival
A	LH-100	10:00 a.m.	**2:00 p.m. Dec 02**
B	LH-100	10:00 a.m.	2:00 p.m.
C	LH-100	10:00 a.m.	**3:00 p.m.**

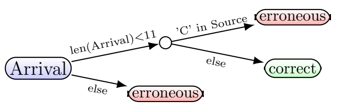

Fig. 3. A trained decision tree to explain the errors of the attribute *Arrival*.

metadata-driven holistic aggregation of error detection results captures more errors in the dataset than off-the-shelf error detection system and aggregation methods from the literature [1].

Our stacking method requires labeled data up to 1% of the dataset size. Currently, we are working towards a new active learning strategy to further reduce the labeling effort [19].

2.3 Error Explanation

State-of-the-art machine learning-based error detection methods, as described in previous sections, achieve very high detection accuracy. However, the user might not only be interested in a high accuracy but she might also be interested in the underlying cause of the corresponding errors and their context. For instance, in case of a field separation issue, outlier detection methods, syntax checkers, and functional dependency violation detection methods might detect that there is indeed an error. Nonetheless, none of these methods tells the user that the error is related to a field separation issue that can be resolved by a specific strategy. As a first step to address this problem, we propose to leverage the extensive work on feature engineering for error detection where features cover information on the attribute, tuple, and dataset level for each data cell, as discussed in the aforementioned sections. This way, we train a classification model, such as a decision tree, to fit the error detection result that the user is interested in exploring. This classification model provides the user with those features that correlate with the corresponding error and therefore gives the user an idea of the context that this error occurs in. Figure 3 shows an example of this approach on *Flights* data. The trained model has learned that the underlying syntax pattern for *Arrival* requires less than 11 characters. This insight hints that *Arrival* has a formatting issue. Furthermore, it found that the source C is unreliable with respect to the *Arrival* entries and in fact, the source C is the actual error cause.

3 Related Work

There has already been a large body of work on data cleaning [4, 6, 23, 5]. Individual subsystems can be plugged as potential subsystems into our estimation and aggregation framework.

Rule-based data cleaning. There is already extensive research conducted on rule-based data cleaning [24]. The Holistic data cleaning method is developed based on denial constraints [4]. This approach considers the generalization of functional dependencies by translating them into denial constraints. A generalization of functional dependencies was also proposed by the Llunatic system [9]. The Nadeef system [6] treats data quality rules holistically by providing an interface for implementing denial constraints and other user-defined functions. Another line of research employs metric functional dependencies and proposes a strategy to choose high-quality minimal repairs [23]. The Katara system [5] is a knowledge base and crowd-driven data cleaning system. It aligns a dataset with available knowledge bases to identify correct and incorrect data values and to suggest top-k possible repairs for incorrect data. Our approach considers variations of the aforementioned traditional cleaning strategies as potential subsystems that can be aggregated inside a generated workflow.

Machine learning-based data cleaning. There is an increasing trend of utilizing machine learning approaches for data integration and curation tasks. The HoloClean system combines qualitative and quantitative repair signals in a statistical model that allows it to repair erroneous data values [25]. However, HoloClean requires manual hyperparameter optimization and follows a one-shot aggregation strategy. GDR uses active learning to choose the correct update suggested by user-defined functional dependencies [33]. Continuous data cleaning leverages classification to trade off repairing constraints against repairing the data [30]. Ideally, it is desirable to adopt these approaches for a more general cleaning pipeline beyond rule-based data cleaning. Other systems, such as SCARE [32] and ERACER [18] leverage probabilistic models to repair data and assume that all errors can be corrected without human involvement. Finally, ActiveClean cleans the training data for a machine learning application and requires the user to specify how to clean and how to featurize the dataset [14]. In our work, we apply machine learning techniques to build a cleaning workflow orchestrator that learns from cleaning tasks in the past and proposes effective cleaning workflows for a new dataset.

4 Conclusion and Future Directions

We presented our vision and initial steps for supporting the user in building complex pipelines of automated data cleaning tools. Using various machine learning techniques, we aim at leveraging knowledge about cleaning tasks from the past and data profiling to propose cleaning workflow for a new dataset. So far, we are able to estimate the effectiveness of error detection workflows on a dataset and to aggregate error detection results effectively. Also, we have developed a feature representation that enables effective active learning for error detection. Yet, there are some challenging research directions ahead of us:
Understanding metadata. Our experiments show the benefits of incorporating metadata for various tasks. A principled connection between instances of both concepts, metadata and data quality, is yet to be established. For example, the profiling result about *null* values is an indicator for the completeness of a dataset. However, to detect disguised missing values [21], we would need

different metadata [2]. It is thus essential to establish relationships between the metadata and data quality problems, and use them for data cleaning routines.
Learning to transform data values. We plan to extend our active learning-based example-driven approach from error detection to correction. For instance, we can treat error correction as a translation task that translates erroneous cells to correct cells. Following this idea, we can leverage current advances in statistical machine translation [12].

Acknowledgments

This work has been funded by the following three grants: the German Research Foundation (DFG) under grant agreement 387872445, the German Federal Ministry of Education and Research as BBDC II (01IS18025A), and the German Federal Ministry of Transport and Digital Infrastructure as Daystream (19F2013).

References

1. Abedjan, Z., Chu, X., Deng, D., Fernandez, R.C., Ilyas, I.F., Ouzzani, M., Papotti, P., Stonebraker, M., Tang, N.: Detecting data errors: Where are we and what needs to be done? PVLDB **9**(12), 993–1004 (Aug 2016)
2. Abedjan, Z., Golab, L., Naumann, F.: Profiling relational data: a survey. The VLDB Journal **24**(4), 557–581 (2015)
3. Arocena, P.C., Glavic, B., Mecca, G., Miller, R.J., Papotti, P., Santoro, D.: Messing up with bart: error generation for evaluating data-cleaning algorithms. PVLDB **9**(2), 36–47 (2015)
4. Chu, X., Ilyas, I.F., Papotti, P.: Holistic data cleaning: Putting violations into context. In: ICDE. pp. 458–469 (2013)
5. Chu, X., Morcos, J., Ilyas, I.F., Ouzzani, M., Papotti, P., Tang, N., Ye, Y.: Katara: A data cleaning system powered by knowledge bases and crowdsourcing. In: SIGMOD. pp. 1247–1261 (2015)
6. Dallachiesa, M., Ebaid, A., Eldawy, A., Elmagarmid, A., Ilyas, I.F., Ouzzani, M., Tang, N.: Nadeef: A commodity data cleaning system. In: SIGMOD. pp. 541–552 (2013)
7. Das, S., Doan, A., G. C., P.S., Gokhale, C., Konda, P.: The magellan data repository. https://sites.google.com/site/anhaidgroup/useful-stuff/data
8. Fan, W., Geerts, F.: Foundations of data quality management, vol. 4. Morgan & Claypool Publishers (2012)
9. Geerts, F., Mecca, G., Papotti, P., Santoro, D.: The llunatic data-cleaning framework. PVLDB **6**(9), 625–636 (2013)
10. Hould, J.N.: Craft beers dataset. https://www.kaggle.com/nickhould/craft-cans (2017), version 1
11. Kandel, S., Paepcke, A., Hellerstein, J., Heer, J.: Wrangler: Interactive visual specification of data transformation scripts. In: SIGCHI. pp. 3363–3372 (2011)
12. Koehn, P.: Statistical Machine Translation. Cambridge University Press, New York, NY, USA, 1st edn. (2010)
13. Krishnan, S., Haas, D., Franklin, M.J., Wu, E.: Towards reliable interactive data cleaning: a user survey and recommendations. In: Proceedings of the Workshop on Human-In-the-Loop Data Analytics. p. 9. ACM (2016)

14. Krishnan, S., Wang, J., Wu, E., Franklin, M.J., Goldberg, K.: ActiveClean: interactive data cleaning for statistical modeling. PVLDB **9**(12), 948–959 (2016)
15. Mahdavi, M., Abedjan, Z.: Reds: Estimating the performance of error detection strategies based on dirtiness profiles. In: SSDBM (2019)
16. Mahdavi, M., Abedjan, Z., Castro Fernandez, R., Madden, S., Ouzzani, M., Stonebraker, M., Tang, N.: Raha: A configuration-free error detection system. In: SIGMOD (2019)
17. Management, M.S.: "data, analytics, and ai: How trust delivers value". MIT Sloan Management Review (2019), http://bit.ly/mit-data-quality, Accessed: 20.03.2019
18. Mayfield, C., Neville, J., Prabhakar, S.: Eracer: a database approach for statistical inference and data cleaning. In: SIGMOD. pp. 75–86 (2010)
19. Neutatz, F., Mahdavi, M., Abedjan, Z.: ED2: A Case for Active Learning in Error Detection. In: CIKM (2019)
20. Ouzzani, M., Hammady, H., Fedorowicz, Z., Elmagarmid, A.: Rayyan -a web and mobile app for systematic reviews. vol. 5, p. 210 (2016)
21. Pearson, R.K.: The problem of disguised missing data. Acm Sigkdd Explorations Newsletter **8**(1), 83–92 (2006)
22. Pit Claudel, C., Mariet, Z., Harding, R., Madden, S.: Outlier detection in heterogeneous datasets using automatic tuple expansion. Technical Report, MIT (2016)
23. Prokoshyna, N., Szlichta, J., Chiang, F., Miller, R.J., Srivastava, D.: Combining quantitative and logical data cleaning. PVLDB **9**(4), 300–311 (2015)
24. Rahm, E., Do, H.H.: Data cleaning: Problems and current approaches. IEEE Data Eng. Bull. **23**(4), 3–13 (2000)
25. Rekatsinas, T., Chu, X., Ilyas, I.F., Ré, C.: HoloClean: Holistic data repairs with probabilistic inference. PVLDB **10**(11), 1190–1201 (2017)
26. Schelter, S., Lange, D., Schmidt, P., Celikel, M., Biessmann, F., Grafberger, A.: Automating large-scale data quality verification. PVLDB **11**(12) (2018)
27. Stonebraker, M., Ilyas, I.F.: Data integration: The current status and the way forward. IEEE Data Eng. Bull. **41**(2) (2018)
28. Verborgh, R., De Wilde, M.: Using OpenRefine. Packt Publishing Ltd (2013)
29. Visengeriyeva, L., Abedjan, Z.: Metadata-driven error detection. In: SSDBM. pp. 1–12 (2018)
30. Volkovs, M., Chiang, F., Szlichta, J., Miller, R.J.: Continuous data cleaning. In: ICDE. pp. 244–255 (2014)
31. Wolpert, D.H.: Stacked generalization. Neural Networks **5**(2), 241–259 (1992)
32. Yakout, M., Berti-Équille, L., Elmagarmid, A.K.: Don't be scared: use scalable automatic repairing with maximal likelihood and bounded changes. In: SIGMOD. pp. 553–564 (2013)
33. Yakout, M., Elmagarmid, A.K., Neville, J., Ouzzani, M., Ilyas, I.F.: Guided data repair. PVLDB **4**(5), 279–289 (2011)
34. Zhou, Z.H.: Ensemble methods: foundations and algorithms. Chapman & Hall/CRC, 1st edn. (2012)