REDS: Estimating the Performance of Error Detection Strategies Based on Dirtiness Profiles

Mohammad Mahdavi
Technische Universität Berlin
mahdavilahijani@tu-berlin.de

Ziawasch Abedjan
Technische Universität Berlin
abedjan@tu-berlin.de

ABSTRACT

Datasets usually suffer from various data quality problems or data errors. At the same time, there are various error detection strategies to detect different kinds of data errors. To effectively detect data errors, the user has to deploy and test multiple error detection strategies. However, evaluating each error detection strategy on a new dataset requires tedious manual evaluation efforts. Therefore, estimating the performance of each strategy upfront is desirable for a more effective strategy selection. In this paper, we propose a new approach to estimate the performance of error detection strategies. Our intuition is that error detection strategies will perform similarly on similarly dirty datasets. We introduce the novel concept of *dirtiness profiles*, which make datasets comparable with respect to their dirtiness. Our experiments show that our system REDS accurately estimates the performance of error detection strategies and, solely based on automatically extracted features, outperforms the semi-supervised baseline.

ACM Reference Format:
Mohammad Mahdavi and Ziawasch Abedjan. 2019. REDS: Estimating the Performance of Error Detection Strategies Based on Dirtiness Profiles. In *31st International Conference on Scientific and Statistical Database Management (SSDBM '19), July 23–25, 2019, Santa Cruz, CA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3335783.3335808

1 INTRODUCTION

Data scientists spend about 80% of their time preparing and organizing data. One of the most time-consuming tasks in the data preparation pipeline is data cleaning. Dirty datasets, i.e., datasets that contain erroneous values, are hard to integrate into data analytics applications. Data errors are introduced into datasets because of various reasons, such as typos, inconsistent handling of data, and extraction errors [16]. Cleaning datasets generally requires the identification of erroneous values [11] and their correction [17].

Here, we focus on the error detection task. In real-world applications, data owners prefer to verify the error detection results manually, as most owners do not trust algorithms to manipulate the data [2]. Furthermore, state-of-the-art data correction approaches significantly benefit from effective error detection [17]. Current error detection tools leverage different techniques, such as detecting outliers or rule violations in data [2]. To effectively detect all data errors, it is often necessary to apply more than one strategy [2, 11]. Union all [2] is a trivial solution that runs and evaluates all available strategies on a new dataset. Unioning all the marked data values results in a large set of potential data errors, including many false positives, making human evaluation infeasible.

From a user's perspective, it is not trivial to decide which error detection strategies to pick upfront. An existing solution runs each strategy on the dataset and then evaluates the output of the strategy on a data sample to select only the effective strategies [2]. This approach effectively estimates the precision of a strategy but requires the user to evaluate the output of different strategies.

In this paper, we address the problem of assessing the effectiveness of error detection algorithms without requiring the user to evaluate their results. Our intuition is that, on *similarly dirty* datasets, the same error detection strategies will perform similarly well. To define similarity based on dirtiness, we introduce the novel concept of a *dirtiness profile*, which is composed of various metadata features. Mapping each dataset to its dirtiness profile, we train models that learn the similarity between previously cleaned datasets and the new dataset. Based on these similarities, our system estimates the effectiveness of existing error detection strategies on the new dataset. We show that it is possible to achieve high accuracy based on features that are extracted *without* any human involvement.

Estimating the effectiveness of error detection algorithms based on their performance on previously cleaned datasets requires us to address the following questions: (1) Which automatically extractable metadata can describe the dirtiness of a dataset? (2) How can we use the dirtiness profile to estimate the effectiveness of error detection strategies?

To address the aforementioned challenges, we make the following contributions:

- We present the extensible system REDS (Section 2), which leverages previously cleaned datasets to estimate the effectiveness of error detection strategies on a newly arriving dataset. REDS is adapted for our configuration-free error detection system *Raha* [11].
- We introduce the novel concept of dirtiness profiles (Section 3), which is the first effort towards representing the dirtiness of datasets by domain-independent features. Our dirtiness profile covers three dataset similarity dimensions: content-based, structure-based, and quality-based. While most of the features can be automatically extracted, optional human-provided features can also be added to the profile.
- We present several experiments (Section 4) that show the accuracy of REDS in estimating the performance of error detection strategies. REDS significantly outperforms the semi-supervised baseline approach.

SSDBM '19, July 23–25, 2019, Santa Cruz, CA, USA
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6216-0/19/07…$15.00
https://doi.org/10.1145/3335783.3335808

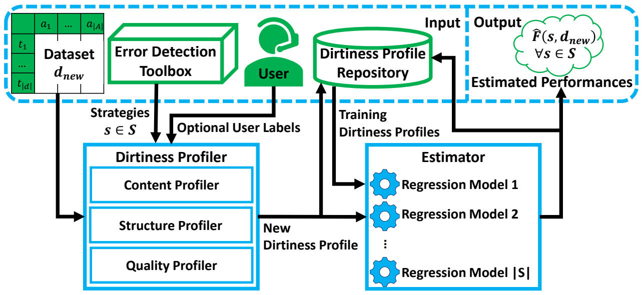


Figure 1: The workflow of REDS.

2 REDS OVERVIEW

Figure 1 shows the workflow of REDS. Given a new dataset d_{new}, an error detection toolbox containing a set of error detection strategies S, a dirtiness profile repository containing the profiles of previously cleaned datasets, and a set of optional user labels, the goal is to estimate the performance (i.e., F_1 score) of all the strategies $s \in S$ on the dataset d_{new}.

To this end, the dirtiness profiler component extracts a set of metadata as the dirtiness profile of the new dataset. The extracted metadata is used by the estimator component to estimate the expected performance of each error detection strategy on the new dataset. The estimator component leverages regression models that have been trained based on historical dirtiness profiles. Note that there are exactly $|S|$ independent regression models, as many as the number of error detection strategies. Each regression model learns to estimate the F_1 score of one specific strategy $s \in S$ on the new dataset d_{new}. In the end, the dirtiness profile of the new dataset along with the estimated F_1 score of strategies are stored in the dirtiness profile repository for retraining the regression models in the future. The contributions of this paper mainly relate to the dirtiness profiler component as it is detailed in the next section.

3 DIRTINESS PROFILE

Intuitively, similar datasets should require similar error detection efforts as well. The similarity of datasets can be defined with regard to different dimensions. Content-based and structure-based similarities are the simplest similarity dimensions for comparing two datasets [1, 15]. However, these similarity dimensions are not enough as two datasets with almost the same content and structure could suffer from different data error types, such as outliers and rule violations. Thus, we argue that, in data cleaning context, *data quality similarity* is another important similarity dimension.

We propose a *dirtiness profile* that summarizes the content, structure, and quality of a dataset into metadata features. Our proposed dirtiness profile has two properties. First, it can be generated for each dataset automatically, i.e., without any user involvement. Second, it contains both domain-dependent (i.e., content) features and domain-independent (i.e., structure and quality) features.

Content features. Content features represent the domain of data. Datasets with similar data domains are likely to have similar data errors as well. For example, multiple datasets in literature, such as *Hospital* [17], *Address* [11], and *Beers* [9], contain data domains *zip*, *city*, and *state*. For these content-wise similar datasets, applying the same rule violation detection strategy that marks data cells violating the functional dependencies such as *zip* $\longrightarrow$ *state* and *city* $\longrightarrow$ *state* as data errors is typically promising. Therefore, to capture the content similarity, the dirtiness profile should leverage features that describe data domains of datasets.

Our dirtiness profile contains the most frequent words and cell values of each data column as features to represent the content of the dataset. This way, datasets with similar data domains (e.g., *city* and *capital*) would have similar dirtiness profiles because of the same overlapping values.

Structure features. Structure of a dataset can be represented via various metadata. Error detection strategies are likely to have similar effectiveness on datasets with certain data value structures. In particular, the distribution of data types is a key characteristic to estimate the effectiveness of error detection strategies. For example, on datasets that mainly contain numerical data values, outlier detection strategies could be more effective than rule violation detection strategies. Therefore, to capture the structural similarity, the dirtiness profile should leverage features that describe the distribution of data value types of datasets.

Our dirtiness profile contains the fraction of unique, explicitly missing, alphabetical, numerical, punctuational, and miscellaneous data values as features to represent the structure of each dataset. This way, datasets with similar distributions of data value types would have similar dirtiness profiles.

Quality features. Datasets with similar data error distribution need similar error detection treatments as well. Ideally, this data error distribution should represent which fraction of each error type exists in the dataset. For example, if we know that 5% of a dataset is outlier and 25% of the same dataset is rule violation, then the rule violation detection strategy is a more effective approach for this dataset rather than the outlier detection strategy.

However, it is not trivial to accurately calculate the distribution of data error types for a given dataset. An existing approach evaluates the output of strategies on a data sample to estimate the distribution of data error types [2]. For example, suppose we evaluate the output of an outlier detection strategy on 1% of a dataset. Suppose 10 actual outlier values are identified by the user on this data sample. Assuming that the dataset has 2500 rows and 20 columns, we can extrapolate our observation to estimate $\frac{10 \times 100}{2500 \times 20} = 2\%$ of data values as outliers. This estimation is expensive as it needs the user to evaluate the output of all the strategies on a data sample. Thus, it would be desirable to have an alternative set of automatically extractable features to represent the quality of datasets.

To this end, we leverage the raw output size and overlap of error detection strategies on datasets. The output size of a specific error detection strategy may correlate with the actual number of data errors. For example, the number of marked data cells by a rule violation detection strategy hints at how many actual rule violations exist in the dataset. Thus, datasets that are associated with similar raw output sizes might be similarly dirty. Let $O_{\text{size}}(s, d)$ be the normalized number of data cells that the strategy s marks as data errors in the dataset d. For all the available error detection strategies $s \in S$, we consider the $O_{\text{size}}(s, d)$ as a feature in the dirtiness profile of dataset d. The output overlap of error detection strategies captures the agreement of strategies on a particular dataset. When two strategies have strongly overlapping raw outputs on a

Table 1: Datasets.

Name	Size	Error Rate	Name	Size	Error Rate
Hospital	1000×20	0.03	Address	94306×12	0.14
Flights	2376×7	0.30	Movies	7390×17	0.06
Rayyan	1000×11	0.09	Restaurants	11840×8	0.01
IT	2262×61	0.20	Soccer	100000×10	0.01
Beers	2410×11	0.16	Tax	200000×15	0.04
Salaries	148647×13	0.01			

Table 2: Error detection strategies.

Name	Configuration	Name	Configuration
Histogram I	0.8, 0.1	Gaussian Mixture I	2, 0.005
Histogram II	0.8, 0.2	Gaussian Mixture II	2, 0.010
Histogram III	0.9, 0.2	Gaussian Mixture III	2, 0.050
Gaussian I	1.0	Partitioned Histogram I	05, 0.8, 0.1
Gaussian II	1.5	Partitioned Histogram II	10, 0.8, 0.1
Gaussian III	2.0	Partitioned Histogram III	15, 0.8, 0.1
NADEEF	All Discovered Rules	OpenRefine	All Discovered Patterns
		KATARA	DBpedia Knowledge Base

dataset, they should also be similarly effective on this dataset. Let $O_{\text{overlap}}(s_\alpha, s_\beta, d)$ be the normalized number of data cells that both strategies s_α and s_β mark in the dataset d as data errors. For all the pairs of strategies $s_\alpha \neq s_\beta \in S$, we consider $O_{\text{overlap}}(s_\alpha, s_\beta, d)$ as a feature in the dirtiness profile of dataset d.

For the sake of completeness, REDS can also incorporate sample precision of strategies as *optional* user-provided features [2]. Our system is able to adequately estimate the performance of strategies without this feature group, as we will show in the experiments. Note that measuring the recall and therefore F_1 score of error detection strategies on a data sample is not possible as we have no information about false and true negatives [2].

4 EVALUATION

We conducted several experiments to evaluate the effectiveness of REDS. We first explain our experimental setup and then discuss the achieved experimental results.

4.1 Experimental Setup

We simulated a scenario where we have several cleaned datasets in our repository with a new dirty dataset arriving. REDS estimates the performance of error detection strategies for this new dataset, according to the well-known *leave-one-out* methodology [18]. We use *mean squared error* (MSE) to measure the quality of the estimated performances as $MSE = \frac{1}{|S|} \sum_{s \in S} \left(F(s, d_{\text{new}}) - \hat{F}(s, d_{\text{new}})\right)^2$, where $F(s, d_{\text{new}})$ is the actual and $\hat{F}(s, d_{\text{new}})$ is the estimated F_1 score of the error detection strategy s on the new dataset d_{new}. In each experimental run, we compute the average MSE by considering each of the datasets as the new dataset. We repeat each experimental run 10 times and report the mean and standard deviation of MSE. As the default parameter setting, we apply all the feature groups and the available dirtiness profiles. We also set the sampling rate for the sample precision features to 1% and the regression models to gradient boosting regression [8]. Our prototype is available online[1].

Datasets. We evaluate our system on 11 diverse datasets as listed in Table 1. *Hospital* [17], *Flights* [17], *Rayyan* [12], *IT* [2], *Beers* [9], *Salaries* [20] are real-world datasets that we have obtained along with their ground truth from previous research projects. *Address* [11] is a proprietary dataset with ground truth. *Movies* and *Restaurants* are available datasets in the Magellan repository [7]. We used the existing labels for the duplicate tuples to provide ground truth for them. *Soccer* and *Tax* are synthetic datasets from the BART repository [3]. The domains of these datasets are different, so that simple domain-dependent similarity measures will not capture the similarity of datasets. Nevertheless, dirtiness profiles should be able to represent the similarity of these diverse datasets.

[1]https://github.com/BigDaMa/reds

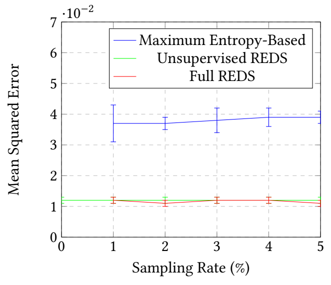
Error detection strategies. We have 15 error detection strategies in our experimental toolbox as listed in Table 2. Histogram Modeling, Gaussian Modeling, Gaussian Mixture Modeling, and Partitioned Histogram Modeling are outlier detection strategies implemented in the dBoost system [14]. NADEEF [6] is a rule violation detection system. OpenRefine [19] can be used to detect pattern violations. KATARA [5] is a knowledge base violation detection system. To configure the rule and pattern violation detection strategies, we used data constraints that were given by the data owners. Additionally, we used the data profiling tool Metanome [13] to discover further valid data constraints on the ground truth of the datasets. The data constraints contain rules, such as functional dependency $zip \longrightarrow city$, and patterns, such as "*zip* should be a 5-digit value". To configure the knowledge base violation detection system, we linked it to the well-known DBpedia knowledge base [4]. For the outlier detection strategies, there is a wide range of possible statistical parameters, which are typically unknown to the user. For this reason, we configured each outlier detection strategy with three recommended parameters based on the dBoost system suggestions.

4.2 Experimental Results

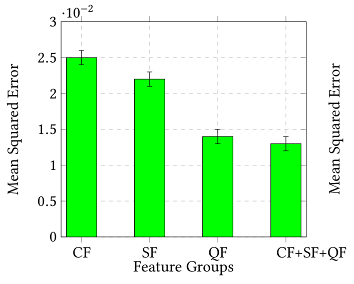
System effectiveness. To show the effectiveness of REDS, we compare three approaches:

(1) **Maximum entropy-based approach [2].** In this baseline approach, the user runs all the error detection strategies on the new dataset. Then, she evaluates the precision of strategies on a data sample. The F_1 score of each strategy is estimated by its sample precision.
(2) **Unsupervised REDS.** In this approach, our system leverages all the features except the optional sample precision of strategies, which need user involvement. We call this approach unsupervised REDS because, here, the dirtiness profile can be generated automatically.
(3) **Full REDS.** In this approach, REDS leverages all the introduced features as the dirtiness profile.

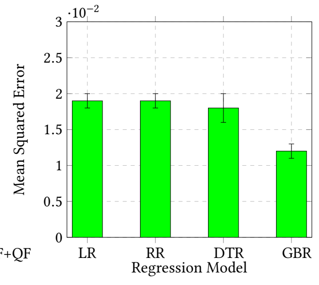
Figure 2(a) illustrates the MSE of the three mentioned approaches. Since the maximum entropy-based approach and full REDS leverage the evaluated precision of strategies on a data sample, we compare them with different sample sizes, i.e., 1% to 5%. The full and unsupervised versions of REDS always outperform the baseline approach. The results particularly show that the unsupervised dirtiness profile works sufficiently effective. Note that the runtime of unsupervised REDS, which does not require human involvement, is in the order of minutes. Extracting those features that are dependent on the output of strategies dominates the runtime as extracting them requires running all the error detection strategies on the new dataset.



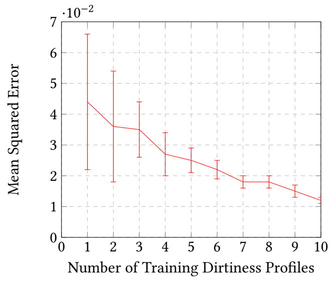
(a) System Effectiveness



(b) Feature Impact Analysis



(c) Model Impact Analysis



(d) Repository Size Impact Analysis

Figure 2: MSE of REDS with different (a) sampling rates, (b) feature groups, (c) regression models, and (d) repository sizes.

Feature impact analysis. Figure 2(b) illustrates the MSE of REDS when it leverages different subsets of feature groups as the dirtiness profile. All the feature groups are informative for the task as REDS estimates most accurately by incorporating all of them (CF+SF+QF). The most informative feature group is data quality features (QF) because it practically captures the error distributions across datasets.

Model impact analysis. Figure 2(c) illustrates the MSE of REDS when it uses different regression models. In particular, we tested linear regression (LR), ridge regression (RR), decision tree regression (DTR), and gradient boosting regression (GBR), all implemented in *scikit-learn* Python module. We applied grid search to find the best hyperparameters for each regression model. The choice of the regression model does not significantly affect the overall performance of REDS as the major performance impact is gained due to the dirtiness profile.

Repository size impact analysis. Figure 2(d) illustrates the MSE of REDS while increasing the number of dirtiness profiles in the repository. We start to train the regression models with just one dirtiness profile. As the number of training dirtiness profiles increases, the MSE decreases as well, because it becomes more likely to find a similar dirtiness profile in the repository. When having around 7 dirtiness profiles as the training set, the performance of REDS almost converges. This is promising as the performance does not require unrealistically large repositories.

5 RELATED WORK

Error detection. Our approach aims at estimating the performance of existing error detection strategies [5, 6, 14]. Maximum entropy-based approach [2] is one of the efforts in this direction that selects strategies based on their precision on evaluated samples. While this approach completely relies on user involvement, REDS learns from previous error detection tasks to omit user labeling costs. Furthermore, REDS includes the sample precision as an optional feature group.

Dataset similarity. Defining the similarity between datasets has been confined to the content-based and structure-based similarities so far [1, 15]. Additionally, we define dirtiness similarity as a novel dimension for measuring the similarity between datasets.

Data repairing. Our approach is orthogonal to data repairing [10, 17] as we are reasoning about error detection. In fact, accurate error detection improves the data repairing procedure as well [17].

6 CONCLUSION

We addressed the problem of estimating the effectiveness of error detection strategies on datasets based on their performance on similarly dirty datasets. We introduced the dirtiness profile that represents the dirtiness of a dataset based on domain-independent features. As our experimental results show, even with only automatically extractable features, our system outperforms the semi-supervised baseline.

ACKNOWLEDGMENTS

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

REFERENCES

[1] Ziawasch Abedjan et al. 2015. Profiling relational data: a survey. *VLDBJ* 24, 4, 557–581.
[2] Ziawasch Abedjan et al. 2016. Detecting data errors: Where are we and what needs to be done? *PVLDB* 9, 12, 993–1004.
[3] Patricia C Arocena et al. 2015. Messing up with BART: error generation for evaluating data-cleaning algorithms. *PVLDB* 9, 2, 36–47.
[4] Sören Auer et al. 2007. Dbpedia: A nucleus for a web of open data. In *ISWC*. 722–735.
[5] Xu Chu et al. 2015. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. In *SIGMOD*. 1247–1261.
[6] Michele Dallachiesa et al. 2013. NADEEF: a commodity data cleaning system. In *SIGMOD*. 541–552.
[7] Sanjib Das et al. 2015. The Magellan Data Repository. https://sites.google.com/site/anhaidgroup/projects/data.
[8] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189–1232.
[9] Jean-Nicholas Hould. 2017. Craft Beers Dataset. https://www.kaggle.com/nickhould/craft-cans. Version 1.
[10] Sanjay Krishnan et al. 2016. ActiveClean: Interactive data cleaning for statistical modeling. *PVLDB* 9, 12, 948–959.
[11] Mohammad Mahdavi et al. 2019. Raha: A configuration-free error detection system. In *SIGMOD*.
[12] Mourad Ouzzani et al. 2016. Rayyan—a web and mobile app for systematic reviews. *Systematic reviews* 5, 1, 210.
[13] Thorsten Papenbrock et al. 2015. Data profiling with Metanome. *PVLDB* 8, 12, 1860–1863.
[14] Clement Pit-Claudel et al. 2016. *Outlier detection in heterogeneous datasets using automatic tuple expansion.* Technical Report MIT-CSAIL-TR-2016-002. MIT.
[15] Erhard Rahm and Philip A Bernstein. 2001. A survey of approaches to automatic schema matching. *VLDBJ* 10, 4, 334–350.
[16] Erhard Rahm and Hong Hai Do. 2000. Data cleaning: Problems and current approaches. *Data Eng. Bulletin* 23, 4, 3–13.
[17] Theodoros Rekatsinas et al. 2017. Holoclean: Holistic data repairs with probabilistic inference. *PVLDB* 10, 11, 1190–1201.
[18] Claude Sammut and Geoffrey I Webb. 2011. *Encyclopedia of machine learning.* Springer Science & Business Media.
[19] Ruben Verborgh et al. 2013. *Using OpenRefine.* Packt Publishing Ltd.
[20] Larysa Visengeriyeva and Ziawasch Abedjan. 2018. Metadata-driven error detection. In *SSDBM*. 1–12.