Estimating the extent of the effects of data quality through observations

Daniele Foroni
Huawei ERC
daniele.foroni@huawei.com

Matteo Lissandrini
Aalborg University
matteo@cs.aau.dk

Yannis Velegrakis
University of Trento and
Utrecht University
i.velegrakis@uu.nl

***Abstract*—Existing data quality works have so far focused on the computation of many data characteristics as a mean of quantifying different quality dimensions, like freshness, consistency, accuracy, or completeness, that are all defined about some ideal (clean) dataset. We claim that this approach falls short in providing a full specification of the quality of the data since it does not take into consideration the task for which the data is to be used, neither any future instances of the dataset. We argue that apart from the difference from the clean dataset, it is equally important to know the degree to which such difference affects the results of the task at hand. Thus, we extend the existing data quality definition to include that degree. Our approach, not only allows data quality to be considered in the context of the intended task, but can also provide useful information even in the absence of the clean dataset, and proffer an understanding of the effect of data quality in future dataset instances. We describe a system and its implementation that computes this extended form of data quality through a principled approach of systematic noise generation and task result evaluation. We perform numerous experiments illustrating the effectiveness of the approach and how this allows contextualizing traditional data quality measures.**

I. Introduction

To have confidence in any data-driven decision making, there should be trust in the data on which the decision is based. Unfortunately, real-world datasets have many data quality issues affecting the results of the analytic tasks applied to them. Consequently, there is an increasing interest on the topic of data quality [9], [13], [16], [18], [19]. Over the years, different data quality dimensions, like accuracy, completeness, consistency, currency, or duplication avoidance, have been identified [7], [27]. The majority of the data quality works, however, has concentrated on the quantification of data quality, to provide an overall informative metric for each dataset [12], [14]. The fundamental principle behind these works is that the available dataset (referred to as the *noisy* or *dirty* dataset) differs from the one that accurately models the reality (referred to as the *clean* dataset). Quantifying data quality translates into quantifying the difference between the noisy and the clean dataset. Unfortunately, the clean dataset is normally not available. A practical solution often adopted is to consider specific characteristics that are expected to hold in the clean dataset, measure the violations of these properties, and consider this as an approximation of the quality quantification [12], [14].

We claim that *simply quantifying the difference between the available and the clean dataset*, as typically done in existing data quality works, *falls short in providing a sufficiently informative indication* of the quality of a dataset. One of the reasons is that this approach *does not take into consideration the task for which the dataset is to be used*. Recall that the main reason we are interested in the quality of a dataset is to know how much trust to put on the results of an analysis task performed on it. Thus, it is not enough to know the difference from the clean dataset, but also how much that difference affects the results of the desired task. It may be the case that the results of a task are highly affected by a small variation from the clean dataset, while the results of another task remain unchanged.

To cope with the above issues, we provide an extension of data quality that takes into consideration the task at hand and is more informative than existing approaches. In particular, we advocate that a more informative metric is the one that encompasses alongside the absolute number indicating the variation from the clean dataset, a factor that indicates the degree in which the results of a specific task are affected by that variation. That factor is bound both to the nature of the dataset and to the specific task. This novel concept of data quality has multiple advantages. First, it allows different data quality values to be specified for the same dataset, but for different tasks. Second, it indicates the degree of error that the results of the task will have on the current dataset, and also on other instances of it. Last, but not least, the factor alone can be used as a data quality indicator that allows to make estimations for unforeseen dataset variations.

We have developed a system that provides a principled materialization of this idea. Given a task of interest, it modifies in a systematic way the available (noisy) dataset by introducing in it various forms of noise, and while it does so, it observes and measures the variation in the results of the specific task of interest applied on the noisy dataset. The many observations are then combined to compute the variation effect factor. To measure the variation in the results of a task, properties specific to the task are used. For instance, if the task is clustering, the Fowlkes-Mallows score [17] may be used to evaluate the effect on the clustering result. In general, any metric can be used.

Empowered by this new type of data quality characterization, analysts may *reason about the reliability and robustness of their insights, prioritize cleaning tasks, or even decide to avoid some of them altogether* [1]. Consider, for instance, an analyst planning to perform some clustering task on a dataset.

37th IEEE Intl. Conference on Data Engineering (ICDE 2021)

The analyst would like to know if it is worth the effort to employ some data cleaning operations, and on what part of the data, or instead, accept the results of the clustering even if they were generated on dirty data. By testing different types of noise, at different parts of the data, the system identifies the effect that the noise has on the results of the clustering. For those that have a significant impact, the analyst can employ the respective cleaning tools to ensure that the obtained results do not differ much from the reality. Moreover, even if they decide not to perform any cleaning, they can monitor the data and, if some kind of noise in the data increases, take action.

The approach taken by our system is in line with other data cleaning systems. Bart [5], for instance, generates different kinds of noise to measure the effectiveness of data cleaning tools. Similarly, ActiveClean [19] by focusing on parts of the data allows for iterative cleaning in statistical modeling problems. While those works mainly focus on cleaning the data, ours gives to the analyst the understanding of the effects of the data quality issues. The analyst can use this knowledge to decide what to clean and when.

More specifically, in this work we make the following contributions: (i) we extend the notion of data quality in a way that takes into consideration the task for which the data is about to be used (Section II). Although it has already been recognized that data quality should be task specific, we are the first to put the task into the data quality evaluation framework formally; (ii) We design a procedure for the systematic computation of the aforementioned task-dependent quality function (Section III); (iii) We materialize the procedure into a fully automated framework that implements different kinds of metrics for different tasks and produces the respective data quality evaluations (Sections IV and VI). (iv) We show how to run our framework for a set of well-known tasks, and we use it to justify for various task evaluation decisions taken elsewhere (Section V); (v) Finally, we perform a number of experiments to evaluate the efficiency of our framework and the effectiveness of our approach (Section VII).

II. Data Quality Revisited

A dataset is a set of structures modeling some real-world situation. Let $\mathcal{D}$ denote the set of all possible datasets.

A data management task, like a query or some data analytics, is a procedure that takes as input a dataset and outputs a set of data structures, i.e., another dataset.

Definition 1: A *task* is a function $T{:}\mathcal{D}{\rightarrow}\mathcal{D}$. The set of all possible tasks is denoted as $\mathcal{T}$.

When a dataset is not accurately modeling the reality, it is said to have *data quality issues*. To distinguish between a dataset that perfectly models the reality and one that does not, we refer to the first as the *clean* and to the second as the *erroneous* or *dirty*.

A *distance function* is a function $d{:}\mathcal{D}{\times}\mathcal{D}{\rightarrow}[0,\infty)$ used to quantify the difference between two datasets.

The *contextual quality* of the dataset D for a task T is a score that depends on the distance between the results of the task when applied on the dataset D, and the results of the same task when applied on the clean dataset D_c, i.e.,

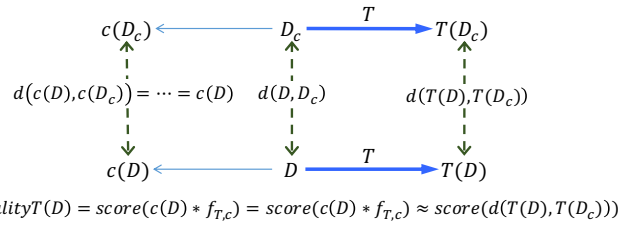

Fig. 1: The Contextual Data Quality Problem Explained

$$Quality_T(D) = score(d(T(D), T(D_c)) \tag{1}$$

where $T(D)$ (respectively $T(D_c)$) denotes the results of the application of the task T on the dataset D (respectively D_c).

A natural assumption often made is that the higher is the distance of a dataset D from the clean dataset D_c, the higher will be the distance of the respective results of a task T on these two datasets, which means that there is a correlation $d(T(D), T(D_c)) \propto d(D, D_c)$. Existing works [10] have adopted this assumption, which has allowed them to quantify data quality by using simply the distance between the clean and the erroneous dataset, i.e., considering the dataset quality as

$$Quality_T(D) = score(d(D, D_c)) \tag{2}$$

Nonetheless, usually the clean dataset D_c is not available, so practical solutions have resorted to compute a proxy of the distance through measuring some data characteristics, the value of which is known for the clean dataset. A *data characteristic* is a function $\mathbf{c}{:}\mathcal{D}{\rightarrow}\mathbb{R}$. For instance, knowing that the clean dataset has no missing (i.e., *null*) values, so the number of missing values in the erroneous dataset is an indication of the distance from the clean one. Denoting the counting function of the missing values as c_{miss}, the accuracy of a dataset with respect to a task T would be: $Quality_T(D) = score(d(D, D_c)) = score(\mathbf{c}_{miss}(D) - \mathbf{c}_{miss}(D_c)) = score(\mathbf{c}_{miss}(D) - 0) = score(\mathbf{c}_{miss}(D))$. Various data characteristics (or combinations of them) can be used to assess different data quality dimensions based on similar assumptions. For instance, a similar approach can be applied to the number of misspellings compared to a fixed vocabulary, or the number of duplicates for an attribute that is supposed to contain unique values.

The limitation of the traditional approach to data quality, which is the one just described, is that it ignores the degree of proportionality between $d(D, D_c)$ and the actual value for $d(T(D), T(D_c))$, which differs from task to task. To cope with this limitation, *the definition of data quality needs to be extended to include it*.

Definition 2: The *quality* of a dataset D for a task T, is

$$score(\mathbf{c}(D) * f_{T,\mathbf{c}})$$

where $score{:}\mathbb{R}{\rightarrow}\mathbb{R}$ is a scoring function, $\mathbf{c}$ is a data characteristic, and $f_{T,\mathbf{c}}{\in}[0, 1]$ is a *sensitivity factor* of the task T to $\mathbf{c}$.

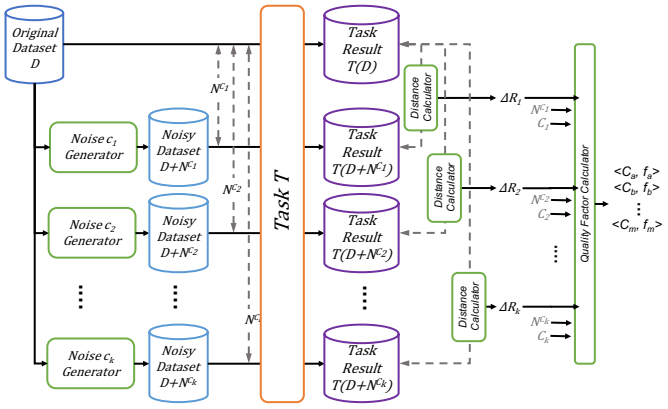

Fig. 2: The architecture of the Contextual Data Quality Evaluation Framework

This characterization of data quality is more informative since it offers a better understanding of the effect of the variation in the data on the task. *Under this definition, an effective assessment of the quality of a dataset for a specific task T requires both the identification of the data characteristics* $\mathbf{c}\in C$ *that are creating large differentiation in the task results and the corresponding computation of the sensitivity factor* $f_{T,\mathbf{c}}$ *for them*. Figure 1 illustrates the theoretical framework for the contextual data quality problem. The challenging task in this process is the derivation of that factor.

III. A Data Quality Framework

To evaluate the quality of a dataset D for a task T, we need to decide what characteristics are worth looking at and how much each affects the task results. To do this, we have developed a framework that tests in a systematic way the effect that a change in the dataset has on the results of the task. Although changes in the dataset can be of any type, we focus on those affecting some specific data characteristic $\mathbf{c}$. We refer to the changes as *noise* and the characteristic $\mathbf{c}$ that they are affecting as the *type* of the noise. For every test performed, the system builds a *scenario*: a triple $\langle \mathbf{c}, N^{\mathbf{c}}, \Delta R\rangle$, where $\mathbf{c}$ is a characteristic, $N^{\mathbf{c}}$ is some amount of noise of type $\mathbf{c}$, and $\Delta R{=}d(T(D), T(D{+}N^{\mathbf{c}}))$ is the change in the results of T when applied to D with and without the noise N^c.

Figure 2 depicts the process performed by the framework and the web site[1] provides a video demonstration on the way it works. Given a dataset, the system produces a series of scenarios by generating some noisy instances with different amounts of noise of the chosen types, and measuring for each noisy instance created the variation in the results of the task from those produced by the given dataset. The framework groups together the generated scenarios by characteristic (i.e., for every noise type) and quantifies the impact on the results of each noise type. The computed degree constitutes the task sensitivity factor $f_{T,\mathbf{c}}$ for that specific type of noise $\mathbf{c}$.

The framework takes as input a dataset D, a task T to apply, a set of noise types $\mathbf{c}$ and the amounts P to introduce in the dataset, a threshold ϵ, and k that represents the repetitions of the process. Since the noise generation can be non-deterministic, running the computation k times enables a more accurate analysis.

First, we apply the task T on the dataset available at hand D, and the result will be used along all the process. Our experiments identified 4 types of behaviors (see Section VII) that represent how the results change with respect to those measured with D while increasing the amount of noise introduced in the dataset: constant, linear, parabolic, and irregular. A negligible impact on the distance (measured by ϵ) characterizes the elements in the first group where no effect is observed in the results regardless of the percentage of noise introduced. A linear relation is observed when, while increasing the percentage of noise introduced, the resulting distance increases too linearly. The parabolic behavior indicates that the distance measured increases with the percentage of noise introduced up to some value, and then it starts decreasing. Finally, the irregular behavior is observed with a fluctuating distance. In Section VI, we expand on how to relate the characteristics of these behaviors to the contextual data quality problem.

Since the framework is aware of the possible behaviors, we can reduce the number of noisy instances of the original dataset to generate for each amount in P when the effect is either constant or parabolic. In particular, the system first generates the noisy instances with the minimum and the maximum noise percentage of P and runs the given task over them. Then, the framework measures the distance of their results from those of the original dataset (Section V). If their distance is lower than the given threshold ϵ, it suggests that the relationship is either constant or parabolic. To discriminate the two, it proceeds by building a scenario for the noise percentages in the middle of the range in P. On the other hand, if the distance is higher than ϵ, we need to consider all the amounts in P and measure their impacts on the results. Then, we calculate the sensitivity factor for the current run with the computed amounts in P. Once we repeated the process k times, we compute the average of the sensitivity factors, which can be implemented in multiple ways (Section VI). Finally, for each noise, we return the sensitivity factors computed.

In what follows, we examine each part of this process, the challenges, and the implementations.

IV. Noise Generators

The noise to produce the different scenarios is generated through the *noise generators*. Each noise generator implements a function $n{:}\mathcal{D}{\rightarrow}\mathcal{D}$ that introduces in the dataset a certain amount of a specific noise type $N^{\mathbf{c}}$. The idea of noise generators is the result of an extensive study of the related literature of benchmark generators, e.g., TPC-H, entity matchers and modifiers, and matching benchmarks [4], as well as many practical scenarios. Following an approach similar to BART [5], a noise generator introduces some noise into the dataset using a given value distribution, e.g., normal and uniform distribution, to allow also biased errors. This nature enables a highly customizable generation of scenarios,

[1] https://db.disi.unitn.eu/projects/f4u.html

handling, for example, the case of an email field that has typos with equal probability in each value and the case of elder people that are more likely to have the birth date missing than young people. Furthermore, we allow the user to specify portions of the dataset in which noise cannot be introduced but, with respect to BART, this noise generator does not handle the conditional application of noise.

The system supports the use of any custom noise generator, but contains already implemented a set of generators for 14 common noise types. These include, among others, the generation of nulls, missing values, spelling mistakes, permuted words, abbreviations, synonyms, terms in different languages, numerical variations, scale modifications and arithmetic negations. It also contains a component that can combine the noises produced by these generators to create noises of more complex types.

V. Measuring Task Result Variations

After having introduced some amounts of a specific noise type in the dataset, the next challenge is to quantify the effect that the noise has on the results of the task, by measuring the difference ΔR between the task results obtained with the original dataset and those obtained with the noisy instance. Different types of metrics can be used for this purpose.

Task-specific Metrics. For many data analytic tasks, there are well-established metrics for quantifying their effectiveness. As an indication, the Fowlkes-Mallows score [17], the Silhouette coefficient, or the Rand Index [25] can be used for clustering, the F1 score, the accuracy, the precision or recall for classification, and the Mean Squared Log Error, or the R^2 score for regression. In these cases, the difference of their scores before and after the introduction of some noise in a dataset gives an indication of the impact of the noise on the task results.

Data Characteristic Metrics. Another approach for quantifying the effect of some noise is to measure the variation of some data characteristics in the results of the analytic task, like nulls, entropy, and value cardinality.

VI. Sensitivity Factor Computation

The last step in the data quality evaluation process is the computation of a score that indicates the relation between the noise and its impact on the analytic results, i.e., the *sensitivity factor* (Definition 2).

We assume as input a series of scenarios, i.e., items of the form $\langle \mathbf{c}, N^{\mathbf{c}}, \Delta R \rangle$, related to a single noise type $\mathbf{c}$. Then, we compute a sensitivity factor for the noise $\mathbf{c}$ by considering the set of $\langle N^{\mathbf{c}}, \Delta R \rangle$ pairs.

We employ 3 different methods to compute the sensitivity factor between the amount of noise introduced in the data and the effect on the task results. The first is the linear regression [20], which identifies a linear relation between the two variables. The second method we employ is the polynomial regression, capturing when the relation is polynomial. In both cases, we obtain a pair, containing the score (i.e., the coefficient of determination, R^2) and the regression coefficient (β) of the relation. The score represents the goodness of the fit of the model and ranges between 0 (bad fit) and 1 (perfect fit). The third method is the Spearman correlation [24] that assesses monotonic relationships, even if they are not linear. It indicates how much the two variables are correlated, and how much they have a common increasing or decreasing monotonic trend. It ranges between -1, meaning negative correlation of the variables, and 1, positive correlation, with 0 meaning no correlation. As described in Section III, we classify these values in 4 sensitivity factor classes: linear, parabolic, constant, and irregular. A high linear regression score and a Spearman correlation close to 1 or -1 characterize the linear group. The parabolic behavior presents a high polynomial regression score (with a lower linear score, otherwise every linear would be categorized as polynomial). A very low standard deviation in the distances and a high Spearman value (close to 1 or -1) distinguish the constant class. Finally, the irregular pattern shows a low linear and polynomial score and a high standard deviation value.

VII. Experiments

In this section, we study the different aspects of our framework, and we demonstrate how it can help analysts to understand the extent to which a particular data quality issue affects the results of a given analytical task. We showcase that the impact of a data quality issue on the results of a task depends on several factors: the type of the noise, its amount, and, most importantly, the task.

In what follows, we demonstrate: (i) how to use our framework to put data quality in context; (ii) the ability of the framework to help the analysts to identify which data quality issues affect more the results of the analytical task at hand; (iii) the scalability of our system.

System Description. We performed a number of experiments to study the different aspects or our system that is implemented in Python and Spark. As analytic tasks, we considered clustering (k-means), classification (Random Forest), and regression (Linear Least Square), well studied tasks with some established evaluation metrics for comparing and understanding the quality of the output. We used the AIRLINES dataset [22]. We run on it 10 noise generators with 10 different percentages $(5, 10, 20, \ldots, 90\%)$, in a 5-fold fashion and report the average.

Using the system. To understand how the system is used by the analysts and the benefits it brings, consider a classification task over the `AIRLINES` dataset. First the system introduces noise of the different types into all the attributes to obtain a set of different noisy instances of the dataset. Then, the Random Forest Classifier is used first over the original dataset, then over the generated noisy instances. The distance between the results is obtained by comparing their F1 scores. Figure 3 (first row, second column) reports the results produced by the system. The closer the F1 on a noisy instance is to the F1 on the original dataset, the lower is the effect of that particular noise on the task, and vice-versa. The plot shows that every type of noise affects the results (as expected). Yet, `MISSING`

INFO and NULL errors have the largest impact on the results. On the other hand, SHUFFLING and ACRONYM do not affect the results significantly since their F1 scores are very similar to the original dataset. As a consequence, one can understand that the time spent by an analyst to clean or repair these latter errors would not bring any significant benefit to the results.

Analysis of complex tasks. We tested the framework with complex data mining tasks that involve both unsupervised (i.e., clustering) and supervised learning (i.e., classification and regression). Figure 3 presents the distances from the results in the original datasets, for all the configurations tested. The distance measure we used are: for clustering the Fowlkes-Mallows (FM) score [17]; for classification the F1 score; and for regression the Mean Squared Log Error (MSLE). For the first two measures 1 means best quality and 0 worst quality, whereas for the last measure 0 reflects a perfect result, while the higher is the value, the lower is the precision of the prediction. The ground-truth classes/clusters are 4 for the AIRLINES dataset.

The effect of noise on different tasks. For the AIRLINES dataset, our framework was able to detect that the same type of noise affects the quality of the results of each task in a highly different way. If we look at the effects of SCALE on the three tasks, we can see that these errors are the most significant for clustering (a decrease of around 70% with 60% of noise), while for classification and regressions, NULL, EDIT, and NEGATION impact the most: NULL and EDIT for classification with 0.36 of F1 with 80% of noise and an MSLE of 1.7 with 50% for NULL and NEGATION in the latter.

If, furthermore, we compare the effects of MISSING tuples on classification and regression, we can see that the latter suffers less than the former (only in classification this type of noise causes the lowest quality). On the other hand, NEGATION errors affect regression much more than classification, producing some of the worst MSLE increases – up to 1.7 –, while causing only a mild decrease of F1 score – down to 0.3 – respectively, both at 50% of noise introduced.

Therefore, from this analysis, we can see that we must consider not only the data quality issues of the dataset but also the task we want to perform, because what we inferred from one specific task or issue does not always generalize to other tasks or noises. Our framework can effectively support the analyst in conducting this kind of reasoning.

Sensitivity Factor. Above we showed that the quality of the results of a task depends on the type of noise in the dataset, its amount, and the task. To quantify the effect of the noise more precisely, our framework computes a set of *sensitivity factors* for each task, dataset, and noise type, correlating the amount of noise to the results of that task on that dataset. Figure 4 reports all the factors computed.

For example, the polynomial slope generated by the NEGATION generator in clustering for the AIRLINES dataset is very high (-1.92), which means that if the dataset contains even a small amount of negations, cleaning those errors immediately affects the results positively. Similarly, MISSING INFO for classification has a linear coefficient (i.e., 0.99) with an extremely high slope (-0.97), which means that repairing just a few tuples can immediately lead to an improvement in the results. On the other hand, ACRONYM for classification is categorized as constant, meaning that cleaning and repairing such errors would not immediately improve the results of the task. We demonstrated that by taking advantage of the sensitivity factors in Figure 4 and the results shown in Figure 3, the data analyst can plan a cleaning process of the dataset at hand. This allows an improvement in the quality of the results of the task the analyst wants to perform (effectiveness), and involves only the types of noises that significantly affect the task (efficiency). This approach, on the one hand, saves money and time for companies, while on the other hand, gives valuable insights into the data.

VIII. Related Work

Many works have stated the importance of data quality in the modern data ecosystem [6], [11], [23]. One direction in data quality is to quantify the quality of the data by measuring different parameters like freshness [16], completeness [15], and accuracy [14]. Many of these techniques have been implemented in the Metanome framework [21]. These methods fall short in providing full information about the quality of the data since they only indicate the values of the specific quality dimensions. Furthermore, they do not take into consideration the task applied to the data, and assume that the clean dataset (or the properties that hold in it) is always known.

Other works considered as data quality indicators the constraint violations, where the challenge is their efficient discovery. Those focus on functional [3] and inclusion [8] dependencies, with or without conditions [2]. Unfortunately, not all the errors that may appear in the datasets violate constraints and constraints do not always cover all the data.

A generic approach in dealing with datasets with low quality is to eliminate the errors. This is known as data cleaning [1]. KATARA [13] is one of the tools that aim at improving the accuracy of a dataset, through the use of a knowledge base. SampleClean [26] is another tool that given a sample of the dataset, learns how to clean the data techniques over the chosen sample and then applies the discovered rules on aggregate query answers. Using an incremental approach, ActiveClean [19] repairs the dataset prioritizing those records that are likely to affect the results. Crowdsourcing may also be used in data cleaning [9] since it efficiently improves the quality of the dataset adding the human knowledge in the loop.

Last, but not least, it may still be possible to get consistent answers to queries over dirty datasets [10], but these approaches are often restricted to specific query types.

IX. Conclusion

We have extended the notion of data quality to consider the analytic task for which the data is intended. Instead of giving an exact quantification, we provide a factor that indicates the effect of the data quality, and not the data quality itself. It does

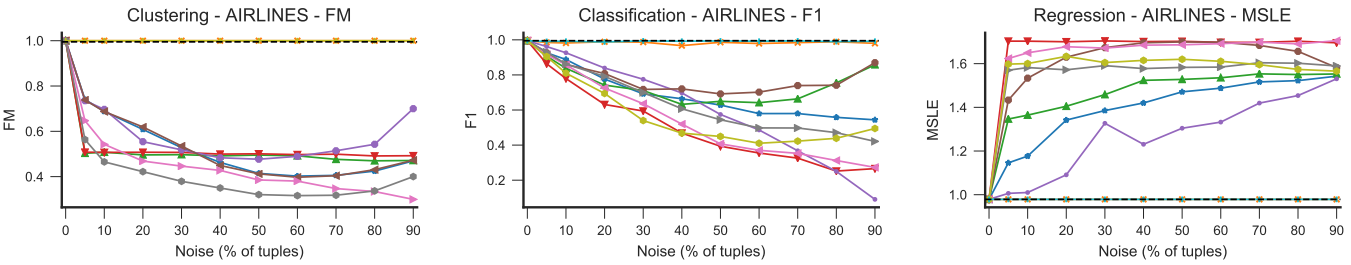

Fig. 3: Task specific distance using FM score for clustering (the higher the better), F1 score for classification (the higher is better), and MSLE for regression (the lower the better)

Task Distance		**Clustering**				**Classification**				**Regression**			
Noise		**Linear**	**Polynomial**	ρ	**Class**	**Linear**	**Polynomial**	ρ	**Class**	**Linear**	**Polynomial**	ρ	**Class**
ABBREVIATION	**AIRLINES**	(0.72, -0.49)	**(0.86, -1.26)**	−0.85*	P	(0.46, -0.31)	**(0.89, -1.39)**	−0.74*	P	**(0.80, 0.48)**	(0.96, 1.25)	0.99*	L
ACRONYM		(1.00, 0.00)	(1.00, 0.00)	0.00	C	(0.01, -0.00)	(0.28, -0.04)	−0.10	C	(0.00, -0.00)	(0.00, 0.00)	0.00	C
BASE CHANGE		(0.25, -0.22)	(0.41, -0.84)	−0.96*	C	(0.02, -0.05)	**(0.99, -1.40)**	−0.15	P	(0.57, 0.37)	**(0.78, 1.17)**	0.98*	P
EDIT		(0.22, -0.20)	(0.39, -0.83)	−0.97*	C	**(0.89, -0.71)**	(0.99, -1.53)	−0.98*	L	(0.17, 0.25)	(0.36, 1.20)	**−0.29**	I
MISSING INFO		–	–	–	–	**(0.99, -0.97)**	(1.00, -0.57)	−1.00*	L	**(0.94, 0.60)**	(0.95, 0.77)	0.98*	L
NEGATION		(0.00, -0.03)	**(0.91, -1.92)**	−0.16	P	(0.02, -0.05)	**(0.97, -1.20)**	−0.10	P	(0.00, 0.05)	**(0.77, 2.48)**	0.23	P
NULL		(0.54, -0.39)	**(0.94, -1.57)**	−0.70*	P	**(0.96, -0.81)**	(0.98, -1.29)	−1.00*	L	(0.26, 0.30)	(0.46, 1.25)	0.97*	C
PERMUTATION		(0.66, -0.47)	**(0.79, -1.24)**	−1.00*	P	**(0.93, -0.58)**	(0.99, -1.14)	−1.00*	L	(0.21, 0.24)	(0.40, 1.03)	**0.77***	I
SCALE		(0.33, -0.32)	**(0.73, -1.58)**	−0.61*	P	(0.60, -0.46)	**(0.99, -1.79)**	−0.67*	P	(0.13, 0.19)	(0.37, 1.11)	**−0.22**	I
SHUFFLING		(1.00, 0.00)	(1.00, 0.00)	0.00	C	(0.00, 0.00)	(0.00, -0.00)	−0.04	C	(0.00, -0.00)	(0.00, 0.00)	0.00	C

Fig. 4: The Sensitivity Factor table reports the tuple (score, slope) for the **Linear** and **Polynomial** relations, the Spearman correlation (ρ), and the **Class** that each noise follows (L=linear, P=polynomial, C=constant, and I=irregular).

so by systematically introducing different forms and sizes of noise, and measuring the variation caused by such noise on the results of the analytic task. We have analyzed all the different aspects of the system and have illustrated the sensitivity faction, an additional information the framework offers in guiding, among others, expensive data cleaning operations.

REFERENCES

[1] Z. Abedjan, X. Chu, D. Deng, R. C. Fernandez, I. F. Ilyas, M. Ouzzani, P. Papotti, M. Stonebraker, and N. Tang. Detecting data errors: Where are we and what needs to be done? *PVLDB*, 2016.

[2] Z. Abedjan, L. Golab, and F. Naumann. Profiling relational data: a survey. *VLDB J.*, 24(4):557–581, 2015.

[3] Z. Abedjan, P. Schulze, and F. Naumann. DFD: efficient functional dependency discovery. In *CIKM*, 2014.

[4] B. Alexe, W. C. Tan, and Y. Velegrakis. STBenchmark: towards a benchmark for mapping systems. *PVLDB*, 1(1), 2008.

[5] P. C. Arocena, B. Glavic, G. Mecca, R. J. Miller, P. Papotti, and D. Santoro. Messing up with BART: error generation for evaluating data-cleaning algorithms. *VLDB*, 9(2), 2015.

[6] C. Batini, A. Rula, M. Scannapieco, and G. Viscusi. From data quality to big data quality. *JDM*, 26(1), 2015.

[7] C. Batini and M. Scannapieco. *Data Quality: Concepts, Methodologies and Techniques*. Springer, 2006.

[8] J. Bauckmann, Z. Abedjan, U. Leser, H. Müller, and F. Naumann. Discovering conditional inclusion dependencies. In *CIKM*, 2012.

[9] M. Bergman, T. Milo, S. Novgorodov, and W. C. Tan. Query-oriented data cleaning with oracles. In *SIGMOD*, 2015.

[10] L. Bertossi and J. Chomicki. Query answering in inconsistent databases. In *Logics for emerging applications of databases*. 2004.

[11] L. Cai and Y. Zhu. The challenges of data quality and data quality assessment in the big data era. *DSJ*, 14, 2015.

[12] F. Chiang and R. J. Miller. Discovering data quality rules. *PVLDB*, 1(1), 2008.

[13] X. Chu, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, N. Tang, and Y. Ye. KATARA: A data cleaning system powered by knowledge bases and crowdsourcing. In *SIGMOD*, 2015.

[14] G. Cong, W. Fan, F. Geerts, X. Jia, and S. Ma. Improving data quality: Consistency and accuracy. In *PVLDB*, 2007.

[15] W. Fan and F. Geerts. Capturing missing tuples and missing values. In *PODS*, 2010.

[16] W. Fan, F. Geerts, and J. Wijsen. Determining the currency of data. In *PODS*, 2011.

[17] E. B. Fowlkes and C. L. Mallows. A method for comparing two hierarchical clusterings. *JASA*, 78(383):553, 569, 1983.

[18] I. F. Ilyas, X. Chu, et al. Trends in cleaning relational data: Consistency and deduplication. *Foundations and Trends® in Databases*, 5(4), 2015.

[19] S. Krishnan, J. Wang, E. Wu, M. J. Franklin, and K. Goldberg. Activeclean: interactive data cleaning for statistical modeling. *PVLDB*, 9(12), 2016.

[20] J. Neter, M. H. Kutner, C. J. Nachtsheim, and W. Wasserman. *Applied linear statistical models*, volume 4. Irwin Chicago, 1996.

[21] T. Papenbrock, T. Bergmann, M. Finke, J. Zwiener, and F. Naumann. Data Profiling with Metanome. *PVLDB*, 8(12), 2015.

[22] N. Park, M. Mohammadi, K. Gorde, S. Jajodia, H. Park, and Y. Kim. Data synthesis based on generative adversarial networks. *PVLDB*, 11(10):1071–1083, 2018.

[23] B. Saha and D. Srivastava. Data quality: The other face of big data. In *ICDE*, 2014.

[24] C. Spearman. The proof and measurement of association between two things. *AJP*, 15(1), 1904.

[25] N. X. Vinh, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *JMLR*, 11(Oct), 2010.

[26] J. Wang, S. Krishnan, M. J. Franklin, K. Goldberg, T. Kraska, and T. Milo. A sample-and-clean framework for fast and accurate query processing on dirty data. In *SIGMOD*, 2014.

[27] R. Y. Wang and D. M. Strong. Beyond accuracy: What data quality means to data consumers. *JMIS*, 12(4), 1996.