Università degli Studi di Trento
Department of Information Engineering and Computer Science

International Doctorate School in Information and Communication Technologies
Cycle XXX

Dissertation for the degree of Doctor of Philosophy

Putting Data Quality in Context

How to generate more accurate analyses

Candidate:
Daniele Foroni

Advisor:
Prof. Yannis Velegrakis

Academic Year 2018-2019

Daniele Foroni - *Putting Data Quality in Context*
Dissertation for the degree of Doctor of Philosophy © October 2019.

Advisor:
Prof. Yannis Velegrakis

Examiners:
Alberto Montresor
Paolo Papotti
Radu Tudoran

"Information is not lost in black holes,
but it is not returned in a useful way.

It is like burning an encyclopedia.
Information is not lost, but it is very hard to read."

———

Stephen Hawking,
Higgs boson particle, *The Guardian* (2013)

ABSTRACT

Data quality is a well-known research field that aims at providing an estimation of the quality of the data itself. The research community has, for quite some time, studied the different aspects of data quality and has developed ways to find and clean dirty and incomplete data. In particular, it has so far focused on the computation of a number of data characteristics as a mean of quantifying different quality dimensions, like freshness, consistency, accuracy, number of duplicates, or completeness. However, the proposed approaches lack of an in-depth view of the quality of the data. Actually, most of the works have focused on efficient and effective ways to identify and clean the data inconsistencies, ignoring to a large extent the task that the data is to be used for, avoiding any investment on data cleaning tasks that are needed, while prioritizing data repairing on errors that are not an issue. Nevertheless, for what concerns streaming data, the concept of quality is slightly turned, since it is more focused on the data results than to the actual input data.

Hence, in the context of data quality, we focus mainly on three challenges, highlighting one aspect for each use case. First, we concentrate our attention on the **task** that the user wants to apply over the data, providing a solution to prioritize cleaning algorithms to improve the task results; second, the focus is on the **user** that defines a metric to optimize for a streaming application, and we dynamically scale the resources used by the application to fit the user goal; third, the **data** is at the center and we present a solution for entity matching that focuses on the measurement of a profile of the data that is used to retrieve the similarity metric that gives better results for such data.

The first work concentrates on putting the context of the task that is applied to the data. So, we introduce **F4U** (that stands for **Fitness for Use**), a framework for measuring how fit is a dataset for the intended task, which means how much a dataset leads to good results for the given task. In this system, we take a dataset and perform

a systematic noise generation that creates from it a set of noisy instances. Then, we apply the user given task to the provided dataset and to these noisy instances, and later we measure the difference that the noise has implied in the results of the task, by measuring the distance of the results obtained with the noisy instances compared to those obtained with the original dataset. The distance allows the user to make some analysis on which noise is mostly affecting the results of the task, which enables a prioritization of the cleaning and repairing algorithms to apply over the original data to improve the results. Other works aims at identifying the most prominent data cleaning tools for a dataset, but our work is the first that does it by optimizing the results of the task the user has in mind.

The second work refers to data quality in a streaming context as a goal-oriented analysis for the given task. It is known that streaming data has different requirements with respect to relational data, and, in this context, data is considered of high quality if it is processed according to the user needs. Hence, we build **Moira** on top of Apache Flink, a tool that adapts the resources needed by a query, optimizing the goal metric defined by the user. The optimization enables improvements in the performance for what concerns the metric goal defined for the given query. Before a query is executed, we perform a static analysis that generates the improved query plan, which improves the performance of the goal defined by the user by a different scaling of the resources. The plan is then submitted to Flink and in the meantime a monitoring system collects information about the cluster and the running application. The system systematically creates, accordingly to these collected metrics, a new query plan and systematically checks whether the deployment of the new plan would improve the performance of the given user goal metric.

In the third work, the focus is on the data itself by proposing a solution to a well known problem, entity matching. We propose a framework that gets the insights of the data, by computing the dataset profile. This would be extremely useful to understand what kind of data the system is analyzing, in order to apply the similarity metric that better fits the data. The system has an online and an offline phase. In the offline phase, the system trains its model to find duplicates on the incoming datasets for which the matching tuples are known. Then, the system computes the profile of the dataset by measuring

the accuracy of the results according to multiple similarity metrics. This knowledge would be used in the online phase, where the system divides the records in portions, minimizing the distance of the profile of each portion from the profiles already computed that we know that would lead to interesting results.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

1 INTRODUCTION

We are led to believe that as long as a query or an algorithm works, then we will have the correct answer as output [FG12]. Unfortunately, this may not happen. It is the case of information retrieval, where algorithms can be correct, but the vagueness and the ambiguity of the available information may lead to wrong results. The same happens in databases, for the same reason: in a real world scenario, data are often dirty [IC+15]. It means that we have no warranty to retrieve an answer that is accurate, up-to-date, complete, or even correct, independently from the query or algorithm we used [Bat+15]. Even more so if the data over which we perform the analysis contains errors. These issues highlight the need and the importance of having data of high quality.

In the era of Big Data, data quality has become even more popular [SS14]. The availability of a cumbersome number of sources that generate data is, on the one hand, an advantage, but on the other, it also has some drawbacks. Having access to a larger amount of data enables for both researchers and industries new analysis and algorithms the end user will benefit from in multiple applications. On the other hand, the generated real world datasets are prone to contain many data quality issues, which cost time and money for being cleaned. For example, the U.S. businesses alone spend over $ 3.1 trillion a year in such process [Eck02; IC+15]. Furthermore, the analyses employed may lead to misleading conclusions with erroneous data [OE16]. As an example, training with wrong information a classification algorithm would definitely lower the precision of the model, hence predict an incorrect class for the testing dataset. Thus, all these elements point out again the importance of having data of high quality.

Since the value that data quality brings is undeniable, this topic has been addressed in various fashions, focusing on the different aspects it involves [WS96]. First, it has been considered the *intrinsic* value of the data, hence the quality that a datum has by itself. Second, the

contextual value of the data has been investigated, pointing out the added value of the analyzed data for the actual application. Third, the research on the quality of the data has also focused on the *representational* sphere of the data, considering both the format and the meaning of each record. Finally, the fourth category points the attention on the level of *accessibility* of the data. The concept of quality depicted in this schema is broad, and many research areas do fit in it, such as data profiling [AGN15; Nau13], data analysis and data mining evaluation [Gia+10; LRU14; GKS11], and streaming management and optimization [Han+14; Rus18; Flo+17] among the others.

The enumeration and definition of the distinct aspects of data quality, the so-called dimensions [SS14; Bat+15; WS96], have been key points in the data quality investigation. It goes along with the analysis of the errors and the issues related to the management of a streaming pipeline, as well as the measurement of the effectiveness of the data analysis and data mining algorithms. However, the investigation of these data quality dimensions falls short with the measurement of the quality of a dataset in relation with the task it is intended to be used. To the best of our knowledge, the existing work do not grasp the impact on the application results of the errors in the data, but only the presence of errors or the performance of a cleaning algorithm [MA19]. In addition, the identification of the user needs in a streaming pipeline has not been linked with the allocation of the resources, while the value of the data by itself has received few attention in the entity matching domain [Zhu+18]. Hence, we claim that an important element that in the literature has received few attention is the contextualization of the aspects around the data. Data quality has never really taken into consideration the task with the focus of improving its performance, while a streaming application has never been optimized according to a goal metric defined by the user, and the value that the data has by itself has received few attention in entity matching. On one side, it is plenty of ad-hoc solutions and methodologies to find out the problems and data quality issues related to a pipeline of operations, starting from the data ingestion phase, to the application of an analysis model [Bat+09; Lee+02; Bat+08]. On the other side, these methodologies seem limited for what concerns the reproducibility, and anyway, they do not cover all the areas data quality influences. Moreover, the contextual sphere of data quality

concentrates the effort on the possible definitions of task and on the proposal of methodologies [Bat+09; BST04; Bat+08], which can be misleading for the user given the lack of numerical results. Furthermore, existing works, mainly related to intrinsic data quality and data profiling, evaluate the quality of a dataset by measuring its missing or wrong information with respect to the so-called perfect dataset, which is erroneously considered as always accessible [AGN15].

In this thesis, we propose a user-centric approach to data quality that goes beyond to the traditional works, providing solutions that could be applied not only in a single scenario but to multiple real world application. In particular, we focus on three main topics, which are introduced in the following. First, we place the task at the center of our work, measuring the quality of the available dataset over which the task will be applied as the difference in the results due to a data quality issue (Section 1.1). Then, we focus on the user through MOIRA, a framework for dynamically allocating resources in a streaming system, following the priorities given by the user (Section 1.2). Finally, the data itself is the center of our approach for entity matching that measures the profile of the data for finding the best similarity metrics to apply on the available data, after having learnt such pattern in an offline phase (Section 1.3).

1.1 MAKING DATA QUALITY TASK-AWARE

Over the years, different data quality dimensions, like accuracy, completeness, consistency, currency, or duplication avoidance, have been identified [BS06; WS96]. A number of studies have considered ways to repair the datasets by eliminating the data quality issues [Chu+15; Ber+15], or techniques that are immune to them [FG10a]. The majority of the data quality works, however, has concentrated on the quantification of data quality, in order to provide an informative metric for each dataset [CM08; Con+07]. 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 not always available. A practical solution often adopted is to consider specific properties 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 [CM08; Con+07]. For example, the number of functional dependency violations observed in a dataset can be considered as an indication of its accuracy, as well as the time taken by a user to fix the errors present in the data [KPN15] or the number of inconsistencies to fix for a repaired dataset [Liv+19]. Other characteristics that have been considered in the past include the existence of nulls, missing tuples, the number of duplicates, and the presence of unexpected values [BS06; WS96; Abe+16b].

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 highly 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 a management task performed on it. Thus, it is not only important to know the difference from the clean dataset, but also how much that difference affects the results of the management 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. Another limitation of the existing data quality approaches is that they provide information about the available dataset instance, but say no more about other instances of the dataset. For example, it would have been useful to know not only how many constraint violations are observed in a specific dataset instance, but also what will happen to the results of a management task if this number increases by a specific factor.

To cope with the above issues, we put data quality in context. 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, even in the case in which the clean dataset is known, the factor alone can be used as a data quality indicator, since it allows to make estimations for unforeseen dataset variations.

We have developed a system that provides a principled materialization of this idea. The system may use the existing data quality mechanisms to quantify the quality of a dataset as before. But its main novelty is in computing a factor of the effect of the variation. Given a task of interest, it modifies in a systematic way the 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 [FM83] may be used to evaluate the effect on the clustering result. In general, any metric can be used. For the cases in which the task is some complex analytic computation, with no established metric known or given, we have devised a generic measure between two task results that is based on the Hungarian Algorithm [Kuh55] and can quantify the difference between two task results.

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 [Abe+16b]. Consider, for instance, an analyst planning to perform some clustering task on a dataset. 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. Furthermore, as the degree of impact is now known to the analyst, even if she decides not to perform any cleaning, she can monitor the data and, if in the future 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 [Aro+15; San+16], for instance, generates different kinds of noise to measure the effectiveness of data cleaning tools. Similarly, ActiveClean [Kri+16] by focusing on parts of the data allows for iterative cleaning in statistical modeling problems, and BoostClean [Kri+17] exploits statistical boost to perform data repairs. While those and other works [Abe+16a] have the main focus on cleaning the data, ours is on giving 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.

1.2 ENHANCING DATA QUALITY TO FIT THE USER NEEDS

The quality of the results should be aware of the needs of the user that performs the analysis. Hence, we move to a streaming scenario that has different needs, before contextualizing data quality on the user. While a data batch has a fixed size, data streams receive data of an unbounded length and process it in real-time. Thus, this real-time need leads to several additional issues.

A data stream is continuous and has no fixed length. Furthermore, the number of elements received in a time window, i.e., the input rate, is prone to vary over time. This fluctuation of the input rate may lead to overallocation or underutilization of the system resources since the amount of data to process is changing as the input rate fluctuates. Data Stream Processing (DSP) applications, i.e., queries or analytics over the data, are commonly represented as a directed acyclic graph (DAG), where the nodes are the operations to be performed, and the edges serve as data streams. Hence, the needed optimizations have to be performed on the topology generated by the user query. In particular, using the example of Apache Flink, we present a topology

optimization that this streaming framework enables by default. Each node of the DAG can be replicated several times to parallelize the task operation through the cluster where the application is deployed for improving the performance of the user query. Furthermore, one node can be chained with the succeeding operator, which means that they will be executed on the same machine and in the same thread, avoiding thread-to-thread handover. The greedy approach would be to parallelize each operator as much as possible and to chain them likewise. However, splitting the work of an operator on multiple machines allocating more resources than those needed has still a cost, and we may want actually to reduce it, or we may keep a fine-grained control of the operators while chaining two or more leads to coarse-grained control of the operators. Hence, the deployment of the topology for the DSP applications is significant in order to allocate the right amount of resources and deploy the best topology.

Several works have been proposed to bind the used resources with the incoming data rate. Dhalion is a system built on top of Heron [Apa19b] that "heals" the running application, enabling the self-regulation of the system, detecting symptoms into the system metrics and applying a number of policies to handle the rescheduling or tuning of the topology [Flo+17]. Elastic Allocator gathers information from the cluster usage and exploits a high resource allocation through a greedy-based algorithm [Han+14]. Another work models the problem of the topology that fits better the incoming data as a Markov decision process, with a model based on Reinforcement learning [Rus18].

However, none of these works consider the user goal, the metric the user needs to optimize, for the intended analytics. What we aim at doing is to focus our work on the user needs in order to let her get the results that she wants. Hence, we introduce Moira, whose name derived by a character of the Greek mythology with decision power on the faith of humans, a dynamic cost estimator system, that through the monitoring of the systems decides the "faith" of the running applications, deciding if a redeployment or a tuning is needed for each application to meet its user-defined goal. A user has to specify the query she wants to perform over the data and a goal, a performance metric she wants to optimize that considers three parameters, i.e., throughput, latency, and cost. Before the deployment to the streaming framework, Moira applies a cost-based estimation of the given

analysis, to optimize its deployment and to get closer to the user goal. Then, after the submission of the application to the streaming framework, the dynamic cost estimator monitors multiple cluster metrics and other data taken from the incoming data and, at every defined interval, it triggers a new cost estimation aware of these parameters. If the built topology does not fit the user requirements and a new and better topology can be deployed, then the running application is replaced with the new topology to avoid wasted computation and optimize the metric defined by the user.

1.3 FOCUSING ON THE DATA ITSELF TO IMPROVE THE RESULTS

The data sources available and, in particular, the data they produce are increasing at a tremendous rate. Hence, multiple datasets take into consideration the same entities. For example, a researcher may scrape the Internet to build a dataset about all the restaurants in Europe, while a company like TripAdvisor already has his own dataset regarding food places. Moreover, a dataset about the products stored in a warehouse may contain a couple of times the same object, due to extra erroneous entries to the storage. Thus, repetitions in the same dataset or multiple datasets are a constant issue that data analysts have to deal with daily.

The problem of entity matching has received a lot of attention with the aim of providing the analysts a tool to discover and fix the duplicated entries within a single table or across multiple datasets. The literature presents in particular solutions to speed up the process or increase the quality of the discoveries. Specifically, neural networks have started to be taken into consideration for this purpose [Mud+18]. Moreover, among the many tools available, lots focus on the solution of a single case, applying custom matching rules or involving crowdsourcing or domain expert intervention.

Although the frameworks shown in the literature help the users with improved prediction performance or with lower running time, they are mainly linked with the knowledge of the scenario where they are applied. For example, despite some solutions provides a

way to avoid (or limit) it [Sin+17], the expertise of domain experts is usually needed to decide the similarity measures to apply. Moreover, multiple scenarios compute the results for a bunch of similarities, and then they merge with a custom function their outputs. This approach leads to a waste of time since often, some metrics would not give remarkable results, and hence, their computation should be avoided.

This third topic points out the benefits of the analysis of the data itself for entity matching. Hence, we present our approach that dynamically profiles the portions of the dataset to apply in parallel to them the correct similarity metric, for enhancing the precision of the results. The approach is similar to apply an horizontal partitioning over the dataset, where each partition is compared with a specific similarity metric. First, the framework performs an off-line phase where the model is trained, computing the profile of multiple datasets, and learning the behavior of the different metrics with datasets having diverse profiles. Then, in the on-line phase, the framework takes a dataset and splits it into chunks and applies the metric that the model has learned to be the best for the profile of the analyzed fragment. The results of the pieces are then forwarded to the framework that outputs the dataset with the discovered duplicated tuples. Although the definition of the best clusters is still an open problem [KBI18], clustering is a related technique, where clustering takes the role of blocking, by grouping together similar entities and then comparing only those in the same group.

1.4 OUTLINE

The remainder of this thesis is organized as follows. In Chapter 2 we present an overview of the state of the art and of the related work in the area of data quality and its related fields.

Chapter 3 introduces the problem of considering the task in a data quality evaluation framework, starting from the revisited model of data quality we propose, to an evaluation of the performance of the system and the impact of many data quality issues in the results of several tasks.

Chapter 4 formalizes the problem of data quality related to the user goal, which represent her needs, in the context of allocating the right amount of resources in a distributed streaming environment.

In Chapter 5 we then illustrate how we solved a problem related to entity matching, for providing the user the similarity metric that better fits the available data. We provided an offline system that learns such relationships and the applies them to the incoming data.

Finally, in Chapter 6 we discuss the contributions and the limitations of our approaches, highlighting the future research directions that should be followed, and we conclude with the final remarks.

2 STATE OF THE ART

In this chapter, we provide an analysis of the literature of the main topic of this dissertation, data quality, and the work related to our applications for extending the actual concept of data quality. First, we survey the current studies on data quality, adding a contextualization view into the data quality framework provided in the literature [WS96] (Section 2.1). We then present a broad description of the aspects it involves. The first of them is the intrinsic data quality (Section 2.2), which considers the characteristics that the data has by itself. The second focuses on the contextual value of the quality of the data (Section 2.3), where are presented the works that aim at not focusing only on the intrinsic value of the data, but also investigate the context around the data. Then, the third is about the methodologies that have been proposed in the literature for the assessment and the measurement of the quality of the data (Section 2.4).

For what concerns the applications and studies we present in this thesis, in this chapter, we later focus on their related work, starting from the fitness-for-use idea (Section 2.5), moving to the analysis of the dynamic allocation of the resources (Section 2.6), and ending with a discussion of the most relevant entity resolution work for our approach (Section 2.7).

This overview of the related literature presents the importance that the quality of the data has and has received, and highlights the gaps that we aim at filling with the following chapters.

2.1 INTRODUCTION

Data Quality is the term used to characterize how accurately the data models the reality. Database design principles, schemas, functional dependencies, type restrictions, checks, triggers, keys, referential constraints, and not-null specifications, are all techniques that

have been introduced by database vendors to ensure this. Unfortunately, despite the existence of all these principles and constraints, data quality remains an issue. Since every constraint imposed on the data comes with a price concerning space or time, data administrators may choose not to put all the necessary constraints in place, or the constraints supported may not be expressive enough to prevent any incorrect values from entering the system. The reasons incorrect values are created in the first place is the reality itself. Real life data is often inconsistent, incomplete, duplicated, or obsolete. Different users may have different goals in mind, different styles, or different levels of attention they pay when entering data into the system. Furthermore, faulty equipment may produce missing or erroneous values, data distribution may generate conflicting information, and delays in data update may create outdated values. All these cause a mismatch between the real world and the world as modeled within the system, which means that queries posed on the data generate results that are not reflecting the reality.

Bad Quality is one of the main challenges of modern businesses and is considered the first and main cause of wrong analysis and predictions [SS14; TB98]. The advent of Big Data has only made the data quality assurance more challenging. Big Data is data characterized by large volume, variety, and velocity. Performing any consistency and correction checks before putting such data in the system is practically impossible. Furthermore, Big Data analytics and exploitation have brought a new computational model. Instead of putting the data into predefined structures and ensuring they adhere to rules set by domain experts, the collected raw data is analyzed and form a model of the actual reality (data driven as opposed to the model driven). The above means that big datasets are way more prone to corruption and other quality issues [SS14].

Data quality is measured through a number of different metrics. The metrics range from statistics on the values a data set contains too properties related to the management of these values. A recent study has produced a high-level categorization (illustrated in Figure 2.1) of the different metrics that have so far considered [TB98; WS96]. It was based on two questionnaires that were used to capture the opinion of the data consumers on what is considered data quality [WS96]. It contains four different types of metrics. The first is the so-called *in-*

trinsic and includes metrics that can be computed by considering only the values in the data set and the actual values that describe the reality. Examples of this kind of metrics include the consistency, i.e., how many constraint violations can be observed, how accurate are the values in the dataset, or how recent they are. The second type of data quality metrics is related to the *contextual* quality, which concentrates on the value it add to the context where it is applied. It encompasses metrics that are based on some additional or superimposed information on the data. An example of such a quality metric is the timeless (freshness) or the objectivity. As one can see, to measure the freshness or the objectivity, the data value itself is not enough. There is a need to know when was the last time that this value was updated or who was the person that updated it. The third category is the *representational* that refers to both the format of the data (i.e., if it adheres to a specific standard, if it is concise, etc., and its meaning, i.e., whether the data is easily understandable and interpretable. The last category is the *accessibility* that contains indicators describing what part of the data can be accessed by the users and how easily this can be done. Access methods, security restrictions, path specifications, e.a., are some of the examples of the metrics considered in this category.

The four aforementioned categories can be further distinguished into two groups. We have noticed that the first two are based either on the data itself or on the value they add to the context where it is applied. Instead, the last two categories deal more with issues related to how the data has been stored in the system and the ways that this data can be accessed. We refer to that group as the *system-dependent* group, while to the former as the *system-independent* group. To draw a parallel to the database system design, the first two can be seen as the logical level, while the last two as the system (physical) level. In this chapter, we are not interested in the system dependent quality parameters. We focus on the system independent part, and in particular on the contextual metrics.

There has been a number of surveys on data quality [Bat+09; FG12; Lee+09; LSB15] that focus on general properties the data should satisfy in order to be considered of high quality. The properties are often descriptive without any specific way of turning them into concrete algorithmic and practical data management tasks that one can execute in order to produce and offer some quantitative results. It is

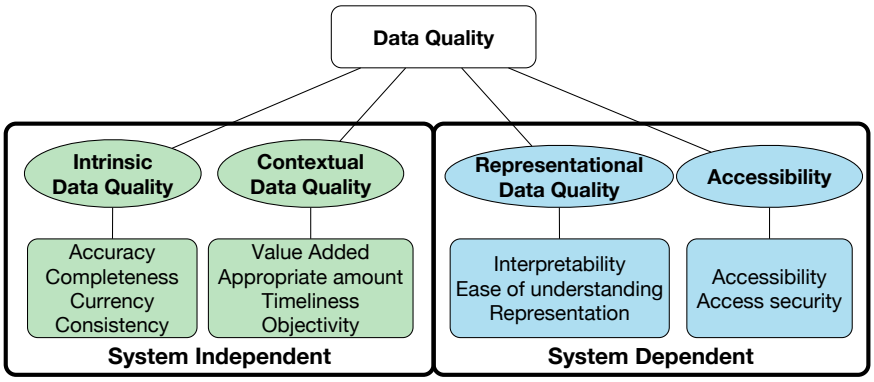

Figure 2.1: The Data Quality Parameters Categorization proposed by [WS96]

often left to the data administrators to translate the generic description into some measurable quantity. On the other hand, there has been a lot of work on data quality metrics that are computed over the dataset. Examples include the counting of the missing values or the constraint violations. They are mainly metrics falling into the intrinsic group and are of particular interest to the data management community due to the many performance challenges. A recent survey [Fan15] has provided a good overview of these metrics, and a book [FG12] from the same authors offers a more in-depth analysis. One of the limitations of these techniques is that they do not consider at all the kind of processing or the intended use of the results of any possible analysis that is to perform on the data. Any possible context information is not part of the input. The only works that the intended use of the data is taken into consideration is the one that is about query answering [Ber11]. There, given an imperfect, i.e., of low quality, dataset and a query at hand, the goal is to ensure that the results of the query are not affected by the low quality of the data.

We believe that any data quality metric should not be seen with respect to the data only but should also consider the context in which that data is about and the kind of analytic tasks it is to be applied on the data, i.e., how the data is going to be used. Based on this position we are aiming in providing a survey on data quality techniques that considered the context where the data is used, but also indicate how those techniques that have not, can be extended to provide a generic

framework. Our work also aims at providing a good understanding of how existing works are limited by not considering the context.

2.2 INTRINSIC DATA QUALITY

We first overview the definition of existing methodologies for the analysis and the improvement of intrinsic data quality features of a dataset. Intrinsic data quality is the aspect of data quality that focuses on the characteristic properties that the data has and that can be measured without considering any external factor. It has been studied thoroughly in the literature, and many dimensions have been categorized as intrinsic [SLW97; WW96; Wan98; Lee+02]. Among them, there is believability, which refers to the credibility of data and how much it can be considered truthful, and objectivity, which defines if the data is unbiased and impartial. However, the most well-known features that are considered as intrinsic to a dataset are accuracy, completeness, time-related dimensions, and consistency.

We describe some works that assess, with a descriptive analysis, and improves, with a prescriptive analysis, the analyzed dimensions. However, according to our knowledge, we have not found any work that considers the predictive analysis, which would forecast how an analysis the outcome would be, given such level of one dimension. This analysis has been more studied on the contextual data quality, and specifically in the methodologies (Section 2.4). We highlight different works on intrinsic data quality in order to present the main features, but we refer to other works and surveys in order to have a deeper overview of the single characteristics of each approach [BS06; Fan15; FG12].

2.2.1 Accuracy

The accuracy dimension has been considered for a long period as a key dimension and as a synonymous for data quality [WS96]. Accuracy has been generally defined as "*the extent to which data are correct, reliable and certified*" [WS96]. More precisely, we agree that data are accurate when the values of the data stored in the database correspond

to the real-world values of the referred object [BP85]. As a result, a formal definition that comes from this statement is that accuracy is the closeness between the correct value v and the represented value v' [Red96].

Two types of accuracy can be identified in the literature: syntactic and semantic accuracy. Syntactic accuracy refers to the closeness of a value v to the elements of the corresponding definition domain, while the semantic captures the cases in which the value v' is a syntactically correct value, but it is different from the right value v. In other words, a value is syntactic inaccurate if the attribute value does not exist or is not accepted, as it is a typo error. Instead, a value is semantically inaccurate if the value is not the right one, as it may happen in a table with a boolean attribute where a tuple contains `false` instead of the correct value `true`. Syntactic accuracy is usually more studied than semantic accuracy as it is easier to be checked [BS06]. Indeed, semantic accuracy requires a knowledge base, a reference dataset, or a human intervention in order to check first whether the data value is correct, and second to replace the value with the correct one if needed.

For this dimension a descriptive analysis would measure the percentage of accurate and inaccurate values in a dataset while a prescriptive analysis would determine what actions, i.e., data repairs, to take in order to improve the accuracy of the dataset. Of those two types of analysis, we found many works that are leverage of users or a knowledge base for a prescriptive analysis [Con+07; Chu+15; Boh+05; CM08; Fan+11], and other works that propose a description of the accuracy of a dataset [FGJ09; Fan+08; CFY13; Rek+15].

A system that uses a knowledge base in order to inspect the semantic accuracy of a dataset and then correct the errors found in the data is KATARA [Chu+15]. This system interprets the input dataset semantic, aligns it with a knowledge base and identifies the values that are correct and those that are incorrect, generating a set of possible repairs for the detected errors. In addition, KATARA involves people and experts in order to resolve ambiguities. Hence, the involved people are asked to select the best repair from those that are suggested by the system according to the aligned knowledge base, and then the most agreed is applied to the data. As a result, this system has better

performance than competitor frameworks, since it takes advantage from the support of both a knowledge base and crowdsourcing.

Another method that has been largely used for the improvement of data accuracy exploits the concept of dependencies, such as functional and inclusion dependencies. Functional dependencies are a constraint that describes the relationship between different attributes in a relation. The standard notation is $A \rightarrow B$, where A and B are (a set of) attributes and the arrow implies that a value for attribute A is associated to only one value for attribute B. Functional dependencies are exploited to discover the primary keys of a dataset. If a primary key exists within a dataset, there is a functional dependency that determines from key values all the other attributes of the relation. For example, if a dataset has three attributes, A, B, and C, with A being a primary key, then there exists a functional dependency $A \rightarrow BC$ that holds. On the other hand, inclusion dependencies imply the existence of attributes within a table, whose values must be a subset of the values of other attributes. Their notation is $R[A] \subseteq S[B]$, where R and S are the same table or two different tables, and A and B are two (set of) attributes with the same cardinality. An inclusion dependency demands that all the values within the attribute (or set of) A of the table R must be present in the attribute (or set of) B of the table S. Inclusion dependencies enable the discovery of foreign keys, since an attribute is a foreign key only if each one of its values belong to the parent table. Thus, an inclusion dependency always exists from the attribute that is a foreign key to the attribute that it references. In the example, $R[A]$ is a foreign key for $S[B]$.

One example of accuracy assessment with dependencies can be found in source integration. In this context, a new technique for database repairing, whose goal is to repair the errors within a dataset with the lowest cost, was introduced taking advantage of functional and inclusion dependencies [Boh+05]. Usually, the database repairing was performed via the insertion and the deletion of tuples, but this technique also considers to repair the dataset with a set of value modifications. The given constraints are used to discover inconsistencies in the dataset, and with a cost function, the value that will be changed at the lowest cost is recognized. This approach improves accuracy since it is able for example to replace the values that are misspelled with the correct values. For example, given a constraint

that does not hold because a set of values are misspelled, if the cost functions estimates as cheaper to repair those values, then the misspellings are replaced with the correct values.

Furthermore, an improvement of standard dependencies that can affect better the data is the concept of conditional dependency. The usage of conditions empowers existing dependencies to recognize more inconsistencies in the data [FGJ09; Fan+08]. A condition is a pattern that imposes the dependency to hold only on the subset of data that match the condition and which combination should occur together [Fan+08]. The standard notation of a condition is $A = v$, where A is an attribute, and v is the value for which the condition holds. It is possible that the value is not specified. In this case, it is assumed that $v = \_$, which means that it can have any value and if it is the only variable in the condition, it works as a normal dependency. A conditional dependency may hold with higher probability than a standard dependency since the condition is imposed on a subset of the data. For example, consider a dataset that stores a set of products to sell, and a functional dependency `[name, tax, country]` $\rightarrow$ `[price]` that does not hold for a pair of tuples. Both the tuples have `France` in the `country` column, while they have different values for the attribute `tax` since one tuple has `tax = 0`, and the other has `tax = 10`. Adding a condition to specify that the dependency holds where `country` is `France` and `tax` is `0`, then the dependency considers only the tuple with `country` as `France` and `tax` as `0`. Thus, the conditional dependency now holds, since the condition is imposed only on a subset of data that does not generate any inconsistency. There exist many types of conditional dependencies and the most known are conditional functional dependencies [CM08; Fan+11] and conditional inclusion dependencies [Bau+12].

Conditional dependencies have been used by many data repairing techniques in order to inspect the accuracy of a dataset, and when applied for database repairing they improve the accuracy of the suggested repairs [Con+07]. This work first finds a possible repair over the constraints that the dataset has, and then uses a statistical method to ensure that the repairs are accurate, without involving too heavily domain experts as it is usually done in the literature. Then, since asking users and domain expert to check each tuple is infeasible, to reduce the number of manual checks, the work employs a sampling

method that selects only a sample of the data that users have to inspect and improve. Thus, this method leverages of an estimation of the accuracy of a repair, which is responsible for the decision of asking or not the user intervention. If the estimated accuracy is lower than a certain threshold, then the repair will be inspected by users, while if it is higher, it will be not.

Another approach focuses on the improvement of the relative accuracy of the data [CFY13] while existing works leverage a reference dataset to check the accuracy of the attributes, this work does not need it. Thus, when many tuples represent the same entity, it finds the correct value for each attribute of the entities. It uses a set of pre-defined rules given as denial constraints to set the value of the attribute. Considering a table with the statistics of a basketball player, a denial constraint over it may state that if the number of matches played of `tuple_1` is lower than `tuple_2`, then `tuple_1` is more current, and thus accurate. Moreover, if the system is not able to compute the repair for the attribute value, then the framework leverages a preference model, which is a scoring function, that produces the top-k possibilities for the replacing value of the entity attribute. Then, users are asked to check among the top possibilities the best one according to them.

Another problem strongly related to data accuracy is the measurement of the accuracy of web data. Given the whole set of data, the accuracy of a web source is measured as the probability of a datum to be in the source and to be correct [Rek+15]. From this definition, they propose a vision system that automatically assesses the overall accuracy of a web source. Such a system chooses which is the best source for the intended use of a user according to its accuracy measure. In order to improve the choice of the best source, other metrics can be added, such as how much the data within the source are up-to-date and unbiased, which is useful for example in the case of news. These two measures can be added to accuracy for the final score computation for the best source.

The measurement of accuracy appears also in the data fusion literature, which refers to the integration of data and knowledge [Cas13]. The problem is about assessing the accuracy of web sources in order to solve disagreements among many of them providing conflicting data that needs to be integrated [Li+12]. This solution measures the

accuracy of a source, considering the true values that it contains over the true value that are present in a reference dataset. Then, some refinements in the measurement are performed considering the standard deviation of the accuracy of a source, which is computed as the difference at each time within a time window between the accuracy of the source at that time and the mean accuracy. In addition, to improve the measurement of the accuracy of a source, the authors suggest to integrate the trustworthiness of the source. However, it is tough to estimate if the information presents in a source is trustworthy and not biased (as it may happen to sources that describe political facts), since the web is evolving fast and the data produced by users may not be correct.

2.2.2 Completeness

Completeness is a broad concept for which there is no single definition on which all researchers agree. In the information system domain exist two main definitions. The first one considers completeness in the intrinsic DQ, and we agree with it since we affirm that completeness is a state of the data and is not affected by the context where it is used. This definition considers completeness as the ability of an information system to represent every meaningful state of a real-world system [WW96]. On the other hand, the second definition focuses on the contextual view of completeness, and as such it will be covered in the next section (Section 2.3). In the domain of data warehouses, completeness has been addressed as the percentage of real-world information entered in a data source or a data warehouse [Jar+13]. Similarly, when we examine entities and attributes in the dataset, it is defined as the information has all the required parts of the description of an entity [BSM03]. Moreover, an enhancement of the last two definitions acknowledges completeness as the product of coverage and density [Nau02]. In this last, coverage considers the number of entities that are present in the dataset with respect to the size of the universal relation, which is the union of all existing data sources. On the other hand, density measures the ratio between non-null values and all the values in the dataset. However, as it can be easily understood, if we are in an open world model, which considers

having no complete knowledge of the world, the universal relation is an ideal dataset. Hence, an objective measurement is difficult to be reached through this definition.

There are three types of completeness: schema completeness, column completeness, and population completeness [PLW02]. Schema completeness regards the degree to which entities and attributes are not missing from the schema; column completeness considers the values that are missing from a column in a table, e.g., null values or missing values; finally, population completeness is defined by means of a reference population and the entities that are missing from it.

For this dimension, a descriptive analysis would measure the percentage of missing data in a dataset, according to other external sources, as in the case of reference data. A prescriptive analysis would determine which would be the repairs for improving the completeness of the tuples and the dataset. One work that is described later has been proposed for improving the completeness of the dataset via a prescriptive analysis [Ber+15] while the others propose both a method to perform a descriptive analysis, which discovers missing entities and attributes, and then provides a prescriptive analysis [FG10c; FG10b].

A new model for handling completeness was introduced for overcoming the problem of dealing with missing values and missing entities [FG10c; FG10b]. Previous works were able to assess the completeness of a dataset either when only tuples [FG10c] or when only values are missing [Van92]. Thus, this work improved upon existing solutions, by making the assumption that a database with complete information exists, which is the reference dataset. The model considers a set of *slave* datasets that satisfy a set of inclusion dependencies with regard to the reference data. If the answer to a query over one of the slaves already contains all possible retrieved entities imposed by the constraints, then the answer to the query is considered complete, otherwise, the query needs to be rewritten, and the data has to be taken from the reference dataset. Thus, the new result set will then contain also the entities that are missing from the slave dataset. This model has lots of applications. However, it needs some prerequisites. Firstly, it needs to have access to a reference dataset containing complete information; secondly, the inclusion dependencies over the

slaves have to be given. Thus, usually, it cannot be applied to any context without the involvement of users or domain experts.

Another work that deals with completeness is a data cleaning framework [Ber+15]. The system receives as input a query and a dataset and produces a draft result set, with the information it contains. Then, as a second step, users are asked to check whether tuples are missing from the draft result set, or whether some tuples need to be removed from it. Finally, extra tuples are removed from the result set, and the missing tuples are added, according to the answers of the users. However, the system does not provide an objective measure of the completeness of the dataset. The involvement of a set of users or, in general, experts are techniques used for completeness assessment and for data quality methodologies, which adopt also surveys to data customers, as shown in Section 2.4.

2.2.3 Time-related dimensions

In the literature are defined three different measures regarding time: volatility, currency, and timeliness. These three concepts of time are used in different ways by different authors, and there is no substantial agreement on the name to use for a time-related dimension. One description of timeliness defines it as the delay between a change of a real world state and the resulting modification of the information system state [WW96], while another work defines currency as the degree to which a datum is up-to-date [Red01]. Timeliness has been also considered as the average age of the data in a source [Nau02]. Only one work defines all three aspects that appear in the literature regarding time: timeliness, currency, and volatility [BSM03]. This work defines timeliness as the combination of volatility and currency. Volatility is the measure of the information instability and hence the frequency of value changes for an entity attribute, whereas currency measures how old the information is, based on how long ago it was recorded. Timeliness has also a contextual viewpoint about the quality, and thus the definition from that perspective is covered in Section 2.3.

A descriptive analysis for this dimension measures how current are the attributes in the dataset [FGW11]. On the other hand, the

prescriptive analysis defines which would be the needed edits for improving the currency of the data [Fan+14].

Different models have been proposed, as in the case of completeness, in order to handle currency inside a dataset [CT05]. The easiest solution is to use temporal databases, which store entities alongside additional metadata, containing the timestamp when each entity was lastly updated. An enhancement of this solution leverages of metadata for each attribute. Thus, each attribute of the entity has a recording containing its last update. Other techniques add metadata for each attribute, which allows entities to store a timestamp for each attribute. An excellent improvement proposes a framework able to handle timely data without timestamps [FGW11]. This framework assumes a set of given constraints, structured as denial constraints, that define the possible evolution of the attributes and the entities. For example, an employee must have worked before being retired, and a woman that is widow must have been married before. Thus, with a given set of constraints, it is possible to handle a dataset containing entities that have evolved during time.

2.2.4 Consistency

In the literature, the definitions of consistency are several, but most refer to the violations that are present within a dataset. One defines it as the dimension that handles the violation of semantic rules defined over (a set of) data items. On the other hand, it has been measured as the number of violation of a given set of integrity constraints over the dataset. The violations are similar to those that regard accuracy, but while for accuracy we deal with them to correct the attribute values, for consistency we need to avoid violations within the dataset, without considering that the data are accurate or not. Each entity of the database must satisfy these integrity constraints, that may involve only one table (e.g., functional dependencies), or multiple tables (e.g., inclusion dependencies).

The discovery and detection of dependencies of a dataset that enable a descriptive analysis of the dataset have been studied accurately [FG12; Con+07; Bau+07; Bau+12]. The consequent improvement of consistency is usually performed via data repairing tech-

niques, that change the dependencies over the dataset or edit the data within the dataset in order to let the dependencies to hold among the dataset. As in the case of accuracy, conditional dependencies are extremely useful for the improvement of consistency [FG12]. The different algorithms implemented for discovering and handling inclusion and functional dependencies with and without conditions are the most efficient way for the creation of constraints, respectively, inter and intra-relations. Moreover, both standard and conditional dependencies are used to deal with the problem of data deduplication, which regards the identification and resolution of different representations of the same real-world entity in the dataset [FG12]. A deep overview of the data deduplication problems and solutions may be found in recent surveys [EIV07; KTR10].

Consistency has been also addressed as the amount of data conflicts that are present in a dataset [RD00; LSB15]. Data conflicts are defined as the deviations that can be found while representing a real-world entity inside a dataset [LÖ09]. Data conflicts can be found either in a single source, when we have only one dataset, and its multiple sources, e.g., in the case of data integration [RD00]. Moreover, they have different behavior when we consider the schema of the data (schema-level) or the data itself (instance-level). The data conflicts that are highlighted by the single-source are mostly derived by missing or wrong constraints over the possible data entries. When we consider schema-less sources, the number of restrictions imposed is usually lower, which leads to a high probability of errors and inconsistencies. Whereas in relational databases, the amount of constraints used, i.e., referential integrity and application-specific integrity constraints, is usually higher and thus the error probability is lower.

At the schema-level, data conflicts occur because of the lack of appropriate model-specific and application-specific integrity constraints, due to data model limitations or poor schema design. Thus, many inconsistencies can be not gathered within the dataset. A solution may be to add other integrity constraints according to the specification of the schema. On the other side, at the instance-level data problems arise from errors and inconsistencies that cannot be avoided at the schema-level, such as misspellings, duplicates, and contradictory values.

The problems of the single-source level are emphasized when multiple sources are integrated, i.e., each source may have different representations of the same data and moreover can contain dirty data. At the schema-level problems arise with different names for the same objects and with structural conflicts, as different structures for the same attribute. Instead, at the instance-level, it is difficult to recognize the same entity in two distinct sources and to deal with a different representation of the same value. However, this is only a data conflict classification that can be used for assessing consistency, but it does not provide any measurement or a solution for inconsistencies.

2.2.5 Other dimensions

Little work has been performed in the literature for other dimensions when compared to the four previously defined, and their assessment is usually proposed only in specific domains. For example, in the geographical and geospatial domain is proposed as dimension the positional accuracy (which is the accuracy of the geospatial position and can be assimilated to accuracy in general, but some works consider them separately) [Goo80; GM13; SFG03]. In the archival domain, the condition of a document stored is considered as a data quality dimension [KW03a; KW03b]. In other domains, other dimensions highly used are believability, which considers how much the data from a source are trustworthy, hugely used in peer-to-peer systems or web data; and objectivity, which refers to the source impartiality in the distribution of information, used mainly in news analysis.

2.3 CONTEXTUAL DATA QUALITY

Contextual data quality is the aspect of data quality that highlights the quality requirements needed considering the context of the task-at-hand. For example, the task of counting the rows of a dataset does not need that the values within each row are accurate, while finding the duplicates within the dataset needs accurate values at least for key attributes. Even in contextual DQ are present many dimensions as in the case of intrinsic DQ. Among them, there are appropriate

amount of data, value added, timeliness, and relevancy. It is worthy to see, as it is shown in Figure 2.1, that the measures belonging to the intrinsic category have a correspondence in the contextual. Completeness as the intrinsic DQ has a reference for contextual DQ as the *appropriate amount of data*, which regards the extent to which data are of sufficient breadth, depth, and score for the task at hand [WS96]. Time dimensions for intrinsic DQ have been mainly considered in the form of currency and volatility. Instead, when contextualized, they focus on timeliness, which is the extent to which data are sufficiently recent or up-to-date for the task to perform [WS96; LC02]. In this section, we do not perform an overview of the corresponding measures of the intrinsic quality, but we report the different ways in which contextual DQ has been assessed. First, we describe the different definitions that have been provided for the concept of context in data applications. Then, we put context in relation to data quality.

2.3.1 Defining Context

Data applications have a strong connection with the context in which they are employs. We grouped the different definitions of context into a unified view to provide a formal definition of context.

Definition 1. A context $C = \langle \{A\}, G \rangle$ is a tuple of a set of attributes $\{A\}$ and a goal G.

An attribute can be a location or a timestamp, and a goal is a high level task, as a company or business goal. The context dynamically affects the data analyzed and represents the environment surrounding the data and the goal of the data customer. Although being often considered only as a description of the environment, the context has been also considered as an active process that has often a significant influence on the way data can be interpreted [Bol+09]. Thus, the context was first defined as the user location [Abo+99], and then it evolves to be represented as a set of attributes that creates a dynamic environment that influences the user behavior [Bol+07a].

The context has been also studied as the environment that is around the application and adding the previous definition it evolves to an ambient that influences the data and the analysis over it [BST04]. Another work relates the definition of context to the choice of the view

of the data that fits better the requirements the context imposes, by tailoring the data schema [Bol+07b]. Thus, a user retrieves different views from the data according to its role and location.

The definition of context can refer to the presentation, location, community, or the user [Bol+09]. When it refers to the presentation, the quality of the data is connected to the ability of the system to adapt the content of the presentation to different channels or devices. It is location-oriented if the data handles with high precision the time and space coordinates. The context refers to the community if there are many relevant variables shared by a group of persons, while is user-centered when it focuses on what the user is doing. In this case, it is referred to the task-at-hand applied over the data.

Context evolution has been also modeled as a Direct Acyclic Graph (DAG), where the nodes with their attributes are the contexts, and the edges denote the changes within the contexts [Cou+05]. For example, an entity may be John, whose role is the traveler, and another is Jane, who is the wife of John, and the context is a train station. The context is represented by a set of entities, in this case, John and Jane. There is a set of roles, which are functions, e.g., John is a traveler, and Jane is the wife of John. The context is represented also by a set of relations that link entities, which can be the position or action that John and Jane are taken, e.g., a face-to-face position while discussing. Finally, we have a set of situations, which is defined by a specific configuration of entities, roles, and relations. Any change in the elements of the context (entities, roles, relations, situations) is an edge between two contexts, which are the nodes. Hence, contexts can be mapped in a DAG and then the goal of the entities, which could be reaching a city with the train, should be analyzed in order to enable a context-aware computation. However, it is trivial to understand that the graph generated by all the possible situations and contexts is extremely complex, and thus it is hard to be represented and managed.

2.3.2 The Context in Contextual DQ

The study of contextual data quality has produced a definition for the data of high quality as data that is *"fit for its use"* defined by

data consumers [SLW97]. Furthermore, the data quality characteristics that are required by an application may change over time, since the requirements change as well. Consequently, having and maintaining data with high quality is hard and implies the continuous tracking and measuring of the requirements and the data characteristics required.

Other works link the context to the organization that has the data and the life cycle of their processes. Thus, many solutions involve data and context experts as the data managers that work in the company. These solutions are called methodologies and are detailed in Section 2.4.

We consider a task any analytical measurement, which can be any data mining algorithm, such as clustering or frequent itemset. Different ways have been addressed to derive the context where the data are used, and we analyze three categories that we define as: implication of dimensions, quality query answering, and metadata usage. These groups are discussed in the rest of this section. However, none of the presented works introduce a quantitative measure that can estimate how good are the data for the applied task, which would enable the prioritization of the cleaning tasks to apply over the dataset to increase the quality of the data for the intended purpose.

2.3.3 Implication of dimensions

The importance of the dimensions may vary changing the context, which in our case is the task. It is easily understandable from the previous example, where accuracy is important for detecting duplicates, while not for counting the rows. This happens for every dimension. Thus, the importance of every dimension varies by changing the task performed, which implies a change in the context.

An analytical framework for a data-driven detection of the dependencies between data quality dimensions has been proposed to derive the most important dimensions in each context [DBB06]. The work starts from standard data quality assessment techniques for many data quality dimensions. Then, for each data attribute, the system inspects with one of the most used metrics in information theory, Shannon's entropy that is particularly suitable for data depen-

dencies [Lin91; Sha51; DBB06], whether the attribute produces any error for each monitored dimension. From the analysis produced by this so-called data quality meta-model, the analytical framework then derives the dependencies between different sets of dimensions classifying them. This work proposes a way to categorize in the correct group among the many existing types the dependency between two sets of data characteristics. Please consider two sets of dimensions $D_1 = \{\texttt{completeness}, \texttt{accuracy}\}$ and $D_2 = \{\texttt{timeliness}, \texttt{consistency}\}$. We have a correspondence between two dimensions when a record contains an error for both the dimensions. A perfect dependency exists when D_1 and D_2 have a one-to-one correspondence of presence of errors, thus when any record that contains an error for a dimension contained in D_1 also contains an error for a dimension contained in D_2 and vice-versa. Two dimensions are independent if a data quality dimension does not give any information about another dimension. There is a partial dependency when D_1 and D_2 have a perfect dependency only between a subset of their dimensions. For example, D_1 and D_2 are partially dependent if `completeness` and `consistency` have a perfect dependency, but there is no dependency among `timeliness` and `accuracy`. Another case happens if D_1 has a larger resolution than D_2, which means that there is a one-way dependency from D_1 to D_2, but at least one dimension of D_2 does not imply an error in a dimension of D_1. In this case, one error for a dimension contained in D_1 implies only one error for a dimension contained in D_2. The degeneration of a D_2 conditioned by one or more data quality dimensions of D_1 means that at least a data quality dimension of D_1 produces an error when an error of D_2 appear. Thus, for example when an error of `accuracy` is always present when every error of a dimension in D_2 happens. The last group is the lower-resolution absolute synonymy, which appears when it is possible to group the dimensions in D_1 and the dimensions in D_2 in such a way that the produced groups generate a perfect dependency between D_1 and D_2. Thus, given $D_{1a} = \{\texttt{completeness}\}$ $D_{1b} = \{\texttt{accuracy}\}$, $D_{2a} = \{\texttt{timeliness}\}$ and $D_{2b} = \{\texttt{consistency}\}$, the subsets D_{1a},D_{2a} and D_{1b},D_{2b} have respectively a perfect dependency. The reconstruction of the dependencies, which means categorizing the relationship between the dimensions, increases the knowledge of dimensions and their relationships.

In order to extend the described analytical framework a conceptual framework has been proposed [BSB10]. The so-called Dependency Discovery in Data Quality (D^3Q) framework can be plugged to any available assessment solution. It leverages a Bayesian network, which does not need a domain knowledge for working, since it models the domain where it is applied, representing the set of dimension analyzed, and their conditioned dependencies without apriori knowledge. The Bayesian network generates a Directed Acyclic Graph where the nodes are the data quality dimensions analyzed and the edges provide their dependencies. The probability of each edge is defined as the prior probability of the dimension of the end node conditioned by the probability of its parent. The analysis of the DAG leads to an understanding of the dependencies between dimensions and provides knowledge about how dimensions impact each other, which is useful to exploit the concept of conditional independence, which happened when there are no edges between two dimensions.

2.3.4 Quality query answering

In the problem of answering to a query with data that does not contain error, context is crucial as it can change the answer provided. Thus, this problem has been linked to the problem of efficiently answer to a query given a set of precomputed views [Halo1]. The definition of a view for every context is present also in database theory and not only in the assessment of contextual data quality.

An approach for retrieving an answer without errors to a query has been proposed in a framework for contextual data quality [BRJ11]. This framework models first a relational schema, which contains the data, for example information about the temperature of a patient and when the temperature was taken. Second, it models additional contextual relational schemas, which are external sources that contains extra information, as it can be information about the nurse that took the temperature measurement. Third, a set of contextual quality predicates defined over the schema, e.g., if it was an oral thermometer, or if the temperature is valid. These quality predicates are used to express the quality requirements requested by data consumers or met by data producers. The quality predicates are defined as views over

the contextual relational schema since they will filter out the data that does not satisfy the imposed conditions from the answers to the queries. It is rewritten adding the contextual quality predicates that are present for the schema. A contextual quality predicate may require an external source for additional information. The framework is also able to manage external sources, and if it is the case, a request is triggered to the external source to check the evaluation of the predicate. Then, the answer to the query is retrieved with only the data that satisfy the quality predicates.

Another solution models a set of dimensions that capture different characteristics of the analyzed context [Bol+09]. Starting from the data, the system builds the Context Dimension Tree, which represents the user information needs. It models the dimensions over the data, displaying the characteristics and the possible values that it can assume. For example, given a company, among the set of all possible dimensions, there can be the role of the user that is querying the data, the time when it performs the request, and the location. Thus, it would create many views of the data based on the role of the user, its location, and the time when the query is asked. The answer is computed at runtime among the views created.

2.3.5 Metadata

Metadata is strongly used in many applications as it can store additional information about the data itself.

One work employs metadata to add information on the best data source for a specific context [CGY]. This work considers a context as an answer to the questions what, when, where, and why. Given a set of sources, the framework clusters items from each different source into groups that have information about similar context. These clusters are called context clusters and can specify information that defines a context on the contained data. One item can be a member of multiple clusters. Given a query, the best source for the context of the query is chosen based on the metadata of each cluster, and the result is returned. Then, the user provides to the system an evaluation of the obtained results. As a final step, the metadata stored for the source and the context cluster are updated giving the user feedback.

The leverage of metadata is also adopted for adding additional information about the user skills [WSE09]. Modeling a task as context-dependent and specifically as user-centric, means that different users have different perceptions of a task, due to their expertize, knowledge, and cultural background. An experiment has been conducted over more than 50 persons, having the goal of planning an advertising campaign with a fixed budget. Users have access to a set of metadata, which contains information about the costs and the impact of many advertising media in multiple geographical locations. In addition, the data provided to the participants contains information about the quality of the data, which can be good, normal, or bad. At the end of the test, users were asked to fill a survey about their expertize in the field and how they receipted the task as ambiguous. Their answers are later used by the researchers that conduct the experiment in order to weight the results according to the expertize and the ambiguity of the task. A person that is not keen in planning an advertising campaign but receives high results probably would not have considered himself as an expert, while a user that marked herself as an expert but receives low results probably is not such an expert. In the end, the work shows that considering both of the two sources of metadata, the first about the task and the quality of the data, and the second about user expertize and ambiguity of the task, provides an improvement in the results than using only the first source or none of them.

2.4 METHODOLOGIES

Methodologies for contextual data quality analysis and improvement are an important instrument that allows data customer to understand and improve the available data. More specifically, a data quality methodology is a set of guidelines and techniques that define a rational process to assess and improve the quality of the data [Bat+09], starting from input information describing a given application context.

2.4.1 Phases of the methodologies

A methodology comprises three main phases, an initial step, which regards the state reconstruction; the assessment phase, where the data quality dimensions are analyzed; and the improvement phase, which regards the changes in the dataset performed to avoid future errors.

The state reconstruction phase focuses on the analysis of the context where the data is applied, gathering information from the organizational processes and the services offered by the company that applies the methodology. Moreover, it investigates how data has been collected, which procedures have been followed, and which data quality problems raised. This phase is, of course, skipped when already present and known.

The second phase concerns the assessment and the measurement of the data quality dimensions that mostly affect the data when performing a specific task. The assessment is focused on gathering information about the data in order to gain a complete overview of the context. This phase contains five steps. First, via an analysis of the data schema and through interviews with customers, the assessment phase aims at understanding the data, the system architecture, and the processes where the data is used. Second, it derives through interviews to data customer the causes that do not allow the data to reach the quality goals. The third step is about creating a model for a process that repairs the dataset from the causes of errors. The fourth step is the measurement of the dimensions that were considered problematic in the second step. The measurement would be done using both objective metrics, which produce a comparable value, and subjective metrics, defined by data users with their perception of the data.

The third and last phase, which is the improvement phase, concerns the development of the model drafted in the assessment phase. As a first step, it is needed to evaluate how much errors and bad data costs to the company, and thus how much wrong are the results produced by analyzing such data. Then, each employee of the company is assigned to a task, considering the involvement of the worker in the process and the management of the data. For example, the employee can be assigned to the control of a particular dimension of the data. The step right after the reassignment of the responsibilities

considers the identification of the causes of the errors and the succeeding identification of all the techniques that would improve the quality of the data. According to the cost analysis already performed, the most effective and efficient technique would be chosen and continuously monitored. The process is continuously adapted to reach the new quality goals updated by data customer. A goal is a quality level that can be lower if there are errors in the data or the process, while they can be higher if any error is found. Once the process is running, a new set of constraints is inserted over the data, and the company organization performs a periodic monitoring.

Not all the methodologies provide an effective solution to the data quality problem. Moreover, some of them only suggest approaches that take into consideration the support of users or consumers. According to our knowledge, the idea of formally measuring the quality of the data within a dataset given a task-at-hand and ranking different datasets for a task to perform is not yet studied nor applied.

2.4.2 Comparison of the methodologies

One of the first methodologies that have been published is the Total Data Quality Management (TDQM) [Wan98], which has been considerably used for planning company activities. This methodology recognizes the sources of the errors present both in the data and in the process that uses the data. It requires as input some extra information about the data, named Information Product (IP). For example, an IP can be a certificate of different types, as a hospital bill, a prescription for a visit, or a certificate of birth [SWZ00]. Then, the quality dimensions that affect the process are defined by the domain experts and are measured over the dataset. Those measures that do not guarantee a high quality standard are selected. The problematic quality measures are taken one at a time and are analyzed, in order to identify which are the causes of the errors. After the process of identifying the causes of low data quality, a set of steps is detected and applied for an improvement in the quality of the data. Thus, the TDQM methodology can be considered an end-to-end process, since it starts from the analysis of the requirements and goes to the implementation of the improvements. Furthermore, it allows the repetition

of the analysis phase to evaluate the improved results [Bat+09]. Thus, if the results are not compliant to the data customer needs or expectations, the entire process can be performed again.

Another methodology that has been extensively studied is called AIM Quality (AIMQ) [Lee+02], which is an objective and domain independent technique that focuses on the measurement of the most known dimensions of a dataset. It is based on the Product and Service Performance model for Information Quality (PSP/IQ), which is represented as a table of four quadrants [KS98]. Each quadrant characterizes the quality dimensions according to product or service delivery and if it conforms to the quality specification or if it meets or exceeds them. On the y-axis, there are two rows: product and service quality. Product quality refers to the dimensions that are visible, such as accuracy and completeness, while service quality considers invisible dimensions, as security and accessibility. On the x-axis there are two columns: conform to specification or meet or exceed specification. Conform to the specification checks if a dimension has reached the goal that data customer addressed, while meet or exceed expectations is about the advantages that the data given to the goal of data consumers. The first step of the AIMQ method is the measurement of a set of dimensions that are given as input. The dimensions that have to be assessed are identified by a first questionnaire submitted to the customer of the company that applies this methodology. Then, via a second survey, each dimension is assigned to a quadrant of the PSP/IQ model, according to the relevancy of the dimension for the task-at-hand. As a final step, the measures are compared with the results of other companies, but any comparison specification or application example is provided in the work.

A third methodology is DQA (Data Quality Assessment) that uses a combination of subjective and objective metrics [PLW02]. It distinguishes between subjective and objective measures, defining a subjective measure as those that are related to the perception and the experience. On the other hand, objectives can be divided into two groups, task-independent, and task-dependent. The dimensions of the former group do not have a contextual knowledge of the application, whereas those of the latter group are developed in specific application contexts, and include business rules and constraints provided by the database administrator. The context defines if a dimension is

subjective or objective. The objective measures are those that are produced using three functions: simple ratio, minimum and maximum operator, which is considered as a single operator, and weighted average. Both the subjective and the objective metrics produce a score, and the two scores are analyzed and compared. The discrepancies between the two measures are identified, and the causes that lead to the discrepancies are determined. Then, the necessary actions in order to avoid the discrepancies are applied over the data. DQA emphasizes the importance of the identification of the causes of errors, but it does not discuss the evaluation of the discrepancies and the changes for the improvement of the quality of the data.

The complete data quality methodology (CDQM) method is another cited methodology [Bat+08; BS06]. It has the goal of being simultaneously complete, flexible and simple, and to this end, it can integrate existing techniques and tools, and it applies to all types of data, structured, semistructured, and unstructured. It comprises three different phases: reconstruction, assessment, and improvement. In the first phase, via interviews and surveys to data customers, it focuses on the reconstruction of the relationships among the organizational units, the processes involved in the company, and the data, for having a complete overview of the company where this methodology is applied. The assessment phase measures a set of data quality dimensions defined by data customers and identifies those dimensions that lead to poor data quality. Then, it sets a new target level for the quality of each dimension and evaluates the costs and the benefits for that improvement by performing an approximate estimation of costs. In the third phase, the best trade-off between the costs and benefits among the different techniques is chosen and applied for improving the quality of the data for the set of measured dimensions.

Another methodology focuses on the assessment phase and the techniques used, rather than providing a complete methodology with also an improvement phase [WBP13]. They propose an Hybrid Approach for assessing data quality that combines the different techniques that are already known. This solution is developed for being used with the different requirements of each organization that would adopt it. The authors performed a full reading of the assessment techniques from the literature, by a peer understanding of the steps that have to be followed by each technique. The results of the reviews are

resolved in a unique set of steps. Then, with extra contributions, as the one provided by the reviewers of the work, they refined the generated list of activities for the assessment techniques. For example, one step may be the definition of data quality requirements, the selection of data items, or the identification of data quality measures. This list is the starting point for the application of this methodology to a company. Thus, in collaboration with the managers of the company that would apply this methodology are specified the requirements for data quality assessment. For example, one requirement can be establishing the actual cost of low data quality. Then, the activities that should be applied to overcome the requirements of the company are chosen from the list of activities. In the example, the activity can be the identification of data quality costs. Finally, the results of the activities are reported to the company, which in our example can be the data quality metrics that cost more.

2.5 QUANTIFYING DATA QUALITY

Many works have stated the importance of data quality in the modern data ecosystem [SS14; Bat+15; CZ15]. One direction in data quality is to quantify the quality of the data by measuring different parameters like freshness [FGW11], completeness [FG10a], and accuracy [Con+07]. Many of these techniques have been implemented in the Metanome framework [Pap+15]. 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 [ASN14] and inclusion [Bau+12] dependencies, with or without conditions [AGN15]. 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 [Abe+16a]. KATARA [Chu+15] is one of the tools that aim at improving the accuracy of a dataset, through the use of a knowledge base. SampleClean [Wan+14] 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 [Kri+16] repairs the dataset prioritizing those records that are likely to affect the results. Although it is close to our goal, this framework does not allow the user to select which errors have to be repaired and which can be skipped, as it may happen in a privacy use-case, where some issues in the data are necessary. Crowdsourcing may also be used in data cleaning [Ber+15] since it efficiently improves the quality of the dataset adding the human knowledge in the loop.

Even in the presence of data quality issues, it may be possible to get certain quality answers from the data. This has led to a number of techniques for guaranteeing consistent answers over inconsistent datasets [BC04]. These approaches, unfortunately, are limited regarding the kind of queries they can correctly answer, and the kind of data quality issues with which they can work correctly.

Existing works that considered the intended task when computing the quality of a dataset [Bat+15; Mer+16] do not tackle the problem of providing a measurement of the quality of the data, and usually need the involvement of a domain-expert in the process. Others have evaluated the impact that the data quality issues along with the task have on the involved the results [Cap+18], but their analysis refers only to a specific task without providing an overall framework. Therefore, none of the existing work provide a holistic and generic methodology to help the analyst understand the relation between the dataset characteristics, its data quality issues, and the task at hand.

2.6 STREAMING RESOURCE ALLOCATION

Streaming analysis has been a trending topic over the last years and many frameworks for such applications have been proposed.

Among them, the most known are Apache Flink [Apa19a], the streaming library of Apache Spark [Apa19d], Apache Storm [Apa19e], and Apache Heron [Apa19b]. Given the number of available tools, one of the main questions on the hype recently is about identifying the best available streaming system. To answer this question, several works compare these frameworks to find the most reliable and fastest system available on the market [Chi+16; PPH16]. Moreover, a positive aspect of these systems is that they allow the deployment of the applications over a cluster of machines, enabling scalable prone analysis. Hence, in this context, resource allocation and how to deploy an application over multiple machines are becoming even more important.

The first investigation track leverages on running multiple applications in the same cloud system, where the resources are limited and the applications have to compete with each other to assure them. From this group, some notable examples are YARN [Vav+13], Mesos [Hin+11], and Abacus [Zha+13]. The basic idea behind these systems is that each application knows the needed resources and the framework takes care of its scheduling and deployment among the nodes of the cluster. However, these systems do not provide an analysis bounded to the application resource allocation during its whole cycle.

There are many works on this side, and among them, Dhalion is a self-regulating system built on top of Apache Heron [Flo+17]. It is a system implemented through a set of Symptom Detectors that check the status of both the incoming stream of data and the allocated resources. Then, the symptoms are used by the so-called Diagnosers, which aim at diagnosing problems from those detected symptoms, e.g., back pressure and over-provisioning. Finally, the performed diagnoses are examined by the Resolvers that take the appropriate decision, e.g., allocate more resources or change the location of a task to avoid back pressure. This analysis continues through all the application running period.

Another approach for enabling elasticity in the deployment of a data stream processing is Elysium [Lom+17]. It is a novel elastic scaling approach that provides efficient resource utilization. Elysium takes advantage of a fine-grained model that estimates the resource usage and enables the independent scaling of the operators and the resources. This allows the system to correctly provision a configuration where the least amount of resources are wasted, through the

application of a prediction module to forecast changes in the amount of elements provided as input, which checks systematically if the current deployment of the resources needs to be scaled, up or down. Moreover, it checks if any operator has to increase or decrease its parallelism, for avoiding operators that act as bottlenecks. Finally, the system measures if the estimated resources are smallest as possible, to avoid resources wasting. These steps are allowed by the monitoring system, that collects the network information and decides the scaling actions.

An application on Apache Storm shows a mechanism to dynamically adapt to the incoming flow [SS18]. The work claim that Apache Storm has not rebalance capabilities at the current state, hence it proposes two main strategies. The first, once a change in the topology is required, as an initial step drains the dataflow in order to avoid message losses. Right after it performs a checkpoint, to ensure that the latest state is saved, and then after the rebalance we restore the most recent state, re-enabling the message reliability only for the checkpoint messages. However, this first approach needs a lot of time in the draining phase. Hence, the second strategy overcomes this issue by capturing the messages that are also in the queue of the tasks, broadcasting the checkpoint event to all the tasks, and then resuming the execution, carefully coordinating all the steps to guarantee consistency and reliability to the system.

An older approach for tuning Hadoop Map/Reduce applications is Starfish [Her+11]. It proposes a self-tuning system that handles the Hadoop configuration without the need for the users to tune it by hand. This automatic tuning can be performed at different granularity levels, from job-level tuning (fine-grained) to workflow-level (coarse-grained) tuning, to fulfill different needs. In addition, the framework presents a language to specify a workload (i.e., a sequence of workflows) along with some metadata, which is added to those automatically gathered from the system in order to improve the configuration performance. Furthermore, this language system works as a recommendation engine for configuring Hadoop applications.

One of the first approaches to elasticity in cloud computing has been employed in IBM for System S [Sch+09]. Such solution permits to an operator or multiple operators to transparently make use of additional resources, CPU cores in this case. The scaling is the response

to a change in the distributed system that can be the presence of another application that requires more resources or an operator that asks for additional computational power. The proposed solution has low overhead and quickly adapts to the new condition. In addition, this approach works even in the presence of multiple elastic operators that reach the best efficiency level.

Elastic Allocator is another adaptive system that gathers information about the cluster both from what concerns the CPU usage and the bandwidth usage [Han+14]. It is claimed to be the first system to use the latter metric for this kind of analysis. The framework is built on top of Apache Storm and it aims at solving the problem of assigning the task operators to the appropriate node of the cluster to improve the performance of the application. It takes the decision, knowing the collected information metrics, through a greedy-based algorithm.

Another work that performs dynamic resource provisioning is named Flower (Flow Elasticity Manager) [Kho+17; KKR17]. The framework collects information from multiple monitoring systems of the cluster at different layers, e.g., data ingestion, analytics, and storage, that are later fit into the control system. This module takes as input the history of the sensor values, which are the measured values and the desired values of each monitored element at a specific time, and dynamically updates the value of the actuator to reach the desired results.

A different approach models the problem as a Markov Decision Process (MDP) [Rus18]. However, usual cases do not have a full knowledge of the system, so the approach exploits a reinforcement learning algorithm to overcome this problem. At each iteration the model checks for each task operator of the application if it has to increase its number of parallel running instances, decrease it, or if the actual value suits the requirements. In addition, it provides an analysis on both a centralized approach, where the system runs in the master node and coordinates all the others, and a decentralized approach, where each node is aware only of the tasks running on it and the parallelism can be increased only on the node's available resources.

DRS (Dynamic Resource Scheduling) is yet another application for dynamic rescheduling the resources assigned to an application [Fu+15].

It comprises two layers, the DRS layer, and the CSP layer. The former contains the monitoring system, which runs the resource optimizer algorithm and performs the actual resource allocation, while the latter is just a framework deployed on top of the streaming processing system. Hence, the CSP (Cloud-based Streaming Processing) layer acts as a middleware to allow the communication between the DRS layer and the streaming system adopted. It proposes a solution for allocating the right amount of resources and assigning them to the right cluster nodes under the constraint of a low-latency application. However, the main point of our solution, the optimization of the metric decided by the user, is not taken into consideration in DRS and in all the other systems proposed in the literature.

2.7 IMPROVING ENTITY RESOLUTION

Entity Resolution is a topic that has received and is still receiving much of attention. It is a fundamental task for the improvement of the quality of a dataset and for increasing the reliability of a data source as well as the analytical results obtained from it [Chr+19]. Entity Resolution has been referred with multiple names, e.g., entity matching, duplicate detection, or record linkage among the others [Wha+09]. This problem focuses on finding different records that refer to the same real-world entity, either from a single source or across multiple sources [KR10].

Ideally, to find duplicates in a dataset with n elements, the user needs to check all the candidates, which are all the possible pairs of records in the dataset, which means n^2 comparisons. The first method for such comparisons is the pairwise-matching that aims at comparing each pair of records to state whether the two records refer to the same real-world entity or not. One of the approaches proposed is based on a set of given rules that can exploit the knowledge of the domain for local decisions [Fan+09]. In a multi-source context, the method starts from the idea of functional dependencies among the columns to suggest the next attribute to consider. Such dependencies, referred to as matching dependencies, have a dynamic semantic and defined based on the similarity metric applied, reduce the er-

rors. Swoosh [Ben+09], instead, takes advantage of the properties of a match and merge function (i.e., idempotence, commutativity, associativity, and representativity) to reduce the number of comparisons performed over a dataset. However, none of these works reduces the comparisons based on the properties that exist between the data.

Another method applied in the literature is the application of classification for understanding whether two records represents the same real-world entity [FS69]. Hence, the system decides autonomously whether there it is a match or a non-match, without the need for domain knowledge to build such a system, which is a strong strategic point. However, the disadvantage of such a method is the creation of the training set, which needs to be filled with lots of positive and negative labeled entries. On this side, Snorkel [Rat+17], a system for building training sets by labeling functions written by the users, could be applied. A comparison for classification and regression trees, against the nearest neighbor approach and an attempt to find the linear combinations that best separate the matches from the non-matches has been performed [Coc+01] and declared the classification base approach as the one that provides the most prominent results.

Distance metrics are a third approach for pairwise matching [EIV06]. These metrics are based on multiple criteria, starting from the actual value of a record, its division in subsequences, the sound used to pronounce it, and many others. Later, a mixture of two or more criteria leads to the development of hybrid approaches. The overall similarity between two objects is usually the weighted average of the similarity between all the attributes considered. Upper and lower bound thresholds are used to indicate possible and not possible matches. With this kind of approaches, the domain knowledge required is limited in the choice of the distance metrics for each involved attribute. As a weak point, there is the need for a meticulous process of parameter tuning for the massive amount of thresholds and weights that have to be fixed.

One of the problems of the pairwise approaches is that the number of comparisons to perform is definitely too high. Thus, due to the high data volume in the Big Data era avoiding useless comparisons has become crucial. To overcome such limitation, the blocking has been introduced. It aims at assigning a record to one or multi-

ple buckets in such a way that the similar tuples are assigned to the same bucket, while records that do not represent the same real-world entity are not. This would avoid useless comparison and would dramatically reduce the number of comparisons to perform. However, the portion of executed comparisons can still be high. This is addressed with an extra step named block processing, which minimizes the number of comparisons, without having any impact on the duplicates found. In this context, the approaches have been classified into three categories: the block building methods, the block processing approaches, and the hybrid blocking techniques.

The block building methods aims at effectively creating the blocks in such a way that each duplicate couple shares at least one block. These methods start from a blocking key (BK), which can be an attribute or a set of them, and, according to this blocking key, they segment the input dataset into multiple partitions (called block) to restrict the subsequent comparisons to the entities belonging to the same block. Standard Blocking [FS69] considers a blocking key and places the elements according to it. If two records have the equal value for the blocking key, then they are placed in the same block, while if the benefits of the two records are different, the two records do not match, and are not placed in the same block. This solution builds non-overlapping blocks and is quite sensitive to typos or noise in general. Therefore, the approach has been further improved to increase the robustness of the framework and enable comparisons even with similar records.

The sorted neighborhood approach [HS95] is another blocking method. It creates blocking keys that are suitable for being ordered, to have similar entities closed to each other. Then, a window of fixed length slides over the records already ordered, and the records within the window are compared and, only if they match, they are placed in the same block. The size window size is the result of a trade-off between the efficiency, due to the time spent for the comparisons, and the effectiveness, in terms of robustness to noise in the blocking key value. Improvements of this solution consider a dynamically adaptive window [Yan+07] and a customizable overlapping degree [DN11].

The second approach is about the block processing methods, which focuses on the deletion of the useless comparisons within a single block. The aim of this approach is the identification of the redun-

dant comparisons, those that have been already executed in at least another block, and those between entities that do not merge. Once identified, these comparisons are removed for increasing the performance and the scalability of the system. A typical example is the iterative blocking [Wha+09], which is based on the continuous analysis done over single blocks. If a pair of records is identified as a match, they are merged and replaced with a new unified entity for all their appearances, even in the other blocks. This is done to avoid duplicate comparisons. Then, the system repeats the checks in all the blocks already processed, trying to exploit the benefits that the merging of the two entities have produced. This approach could also be used with overlapping blocks, but in this case, the repetition of the comparisons within a block after a merge are avoided.

For representing the entities in the real world, which is typically inaccurate and imprecise, collective entity resolution [BG07] presents the relationship between the entities as a graph, where the nodes are the entities, and the weight over an edge is the similarity of the two entities it connects or the representation of their matching. The intuition is that two nodes are more likely to match if they are connected to nodes that represent the same real-world entity. The framework applies hierarchical agglomerative clustering for discovering this behavior, merging the two most similar clusters in an iterative process, until their similarity becomes lower than a given threshold. If the two groups are merged, the entities they represent are merged too. A solution that enables such an approach for large dataset has been proposed [WG13]. The framework applies multiple instances of a black-box entity resolution algorithm, each over a small subset of the data, reproducing a similar blocking technique. The computation can be easily distributed, even if there is the need of waiting for the response of an instance to have the results for another. The first blocks are created using the Canopy Clustering, resulting in having in the same block only entities that are suspected to be the same. Each block maintains the list of its duplicate elements that will be useful for the other blocks. Once the research for duplicates is done, the system returns the set of entities with the merged copies. Starting from collective entity matching, different approaches have been proposed. One [RDG11] considers the entity matching algorithm again as a black-box and defines a small set of entities over which apply it as

neighborhood or cover. The cover creation applies Canopy clustering, and the execution of the distribution has been performed in a Hadoop cluster. With the same paradigm, multi-job optimization has received attention [WC13]. The first technique that the framework proposes enables the sharing of the map output for two jobs that share a subset of the output, while the second technique, enables the sharing of the map input scan and the map output. The main idea of the second technique is that the execution sequence allows the framework to obtain the map output from output already computed.

Finally, the literature reports complete tools that enable multiple steps in the entity resolution process that do not focus exclusively on the scalability. Magellan [Kon+16] is a system thought for guiding the user in all the steps of entity resolution. The experiments show that it is an advanced configurable system that also offers multiple similarity join techniques, compared to the other tools available, and it allows the user to apply a deep learning module, which is a unique feature. Another tool, the JedAI toolkit [Pap+18], enables the application to both structured and semi-structured data, not being tight to the schema of the dataset. It allows the user to create workflows over the entity resolution process steps since it contains a module for both blocking and block processing, as well ass the matching phase. Both these tools can be parallelized but in two different ways. Magellan can be distributed taking advantage of Apache Spark, while the other can be parallelized on multiple cores. These systems provide up-to-date solutions, which however do not consider the structural similarities between the data, e.g. the type of a column, to improve the comparisons.

Similar work to what our approach does focuses on the type of data [Zhu+18]. They claim that existing researchers typically assume that the analyzed dataset contains string-type data, and they apply only a single similarity measure. From this assumption, they create a hybrid approach for a type-based blocking function that takes advantage of varying window size. Different similarity metrics are then applied to the blocks, according to their type, but they do not take advantage of the creation of a profile for the dataset. Instead, for what concerns the similarity application, there are works on understanding what similar means [Wan+11]. They claim that the definition of matching rules is not straightforward. Hence, they aim at identifying

the concept of similarity, by identifying the best similarity functions and thresholds for effectively finding entities from a given dataset. The system proposes techniques for avoiding the computation of redundancies, i.e., combinations of similarities and thresholds that lead to the same conclusion, even in the presence of user preferences, e.g., high precision or high recall, by exploiting a set of matching and non-matching examples. Starting from a broader purpose, the detection of errors in a dataset, Raha [Mah+19] presents a configuration-free system. They state the configuring an error detection framework with different parameters for every dataset is a hard task. Hence, without the need of providing any configuration, Raha assigns a feature vector to each cell of a dataset. The value of each cell would be the output obtained by a particular error detection algorithm with a specific configuration, which is automatically generated by the system. The cells are then clustered together based on their feature vector, and the user has to label one cluster at a time. We apply a similar approach, the creation of a profile, to detect similar records instead of tuples that can contain errors.

3 FITNESS FOR USE - CONTEXTUALIZING THE TASK

In this chapter, we present our work on data quality, in particular on fitness for use. While existing works on data quality aims at finding and measuring what kind and how many errors are present in the data, we propose a new approach that assesses the sensitiveness of the given task for measuring the quality of the data.

Here, we put the focus on the task and we contextualize it proposing a framework for a data quality evaluation that considers also the task. Given a dataset and a task, we first introduce some variations, i.e., noise, into the data and then we test whether and how much the task is sensitive to these changes. We do that even if the dataset is not clean, to have an ideal estimation of the benefits due to the cleaning of the errors. Since the amount of energy, money, and time spent to clean a dataset from multiple errors that may not be present, this analysis would be helpful for the analyst to understand for what kind of errors she should check, enabling a prioritization of the cleaning tasks to apply over the available data.

3.1 CONTRIBUTIONS AND OUTLINE

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 3.3). We consider the data quality of a dataset not as the degree of the noise in it, but as a task-dependent function of the noise. 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. Furthermore, we are not bound to a specific form of data characteristics for quantifying data quality, neither to specific forms of analytic tasks or queries. We are also the

LineID	Client	Location	Destination	Date
A1	Alan	41.891, 12.511	Rome	23 Jan
G8	William	41.891, 12.511	Rome	27 Feb
O3	Bill	41.891, 12.511	Rome	1 Jan
G3	Bob	45.464, 9.191	Milan	23 Jan
M5	Bob	44.493, 2.182	Bologna	1 Jan
S2	Dave	45.464, 9.191	Milan	27 Feb
W7	Carlo	43.604, 5.243	Tolosa	23 Jan
P6	David	44.837, 2.528	Bordeaux	1 Jan

Table 3.1: A fraction of a telecommunication dataset.

first to provide a data quality metric that is not static to the available dataset, but is generic and dynamic allowing quality estimations to be made for future datasets; (ii) We design a procedure for the systematic computation of the aforementioned task-dependent quality function (Section 3.4); (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 3.5 and 3.7). (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 3.6); (v) For complex tasks or when success criteria are not provided, we develop a technique for efficiently and effectively computing differences between datasets and use it to quantify the effect on the task results of the noise in the dataset. Finally, we perform a number of experiments to evaluate the efficiency of our framework and the effectiveness of our approach (Section 3.8).

3.2 A MOTIVATING EXAMPLE

A telecommunication company would like to perform some analytics on a dataset, a fraction of which is illustrated in Table 3.1. It contains information on the location from which the customers of the company called, the destination towards which the call was made, and when. The dataset is suspected to have a number of data quality issues. For instance, there may be, for the same person, different versions of the name (Bill vs. William) or misspellings due to a manual

insertion process. Identifiers are randomly assigned, and there may not be a 1-to-1 correspondence with customers to preserve anonymity. Also, the location could be inaccurate due to the GPS receiver imprecisions. The company knows that the misspellings in the name appear in approximate 5% of the data and that the GPS inaccuracy gives an error of around 5 meters, although in cases there could be more substantial discrepancies when the user is in closed spaces. The company can invest in repairing these errors, by cross-checking with other datasets or combining with data of other sensors. The question is whether it is worth the investment.

The company wants to identify the areas with a high number of calls to install new antennas. Hence, they apply a clustering algorithm on the phone call locations, which means that any misspelling or alternative name in the name field has actually no effect on the results of the clustering, and neither does the anonymization of the calls. This scenario leads the analysts of the company to decide not to invest any resources on the repair of anonymizations and misspellings.

On the other hand, inaccuracies in the locations may end up in misplaced phone calls, and detecting and repairing these inaccuracies is a key task. Nevertheless, the company needs to identify areas at a high granularity, and even though the GPS inaccuracies introduce a few meters diversion from the actual location, the clustering results are not significantly altered. As a consequence, the company analysts can safely put trust in the generated clusters and avoid performing any data cleaning investment. For the analyst to take such decision, the critical information was not simply what inaccuracies exist in the data, but also the fact that for the specific task that she intended to perform was not getting affected significantly by them.

3.3 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 2. A *task* is a function $T{:}\mathcal{D}\to\mathcal{D}$. 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 $d{:}\mathcal{D}\times\mathcal{D}\to[0,\infty)$ 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, i.e.,

$$Quality_T(D) = score(d(T(D), T(D_c)) \tag{3.1}$$

where D_c denotes the clean dataset, and $T(D)$ (respectively $T(D_c)$) 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 [BC04] 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

$$Quality_T(D) = score(d(D, D_c)) \tag{3.2}$$

In most of the cases, the clean dataset D_c is not available, so practical solutions have resorted into a simulation of the distance through measuring some data characteristics, the value of which is known for the clean dataset. A *data characteristic* is a function $c{:}\mathcal{D}\to\mathbb{R}$. 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: $Quality_T(D) = score(d(D, D_c)) = score(c_{miss}(D) - c_{miss}(D_c)) = score(c_{miss}(D) - 0) = score(c_{miss}(D))$.

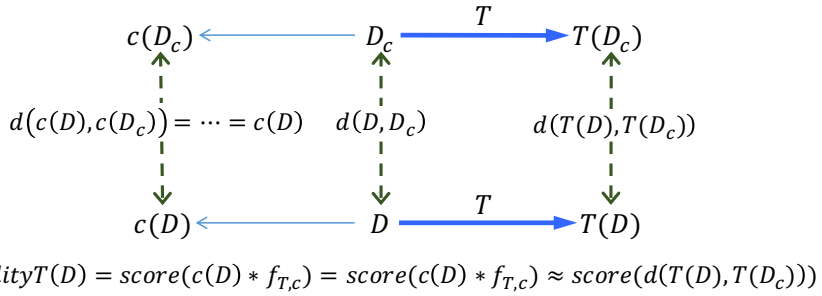

Figure 3.1: The Contextual Data Quality Problem Explained

Various data characteristics (or combinations of them) can be used to assess different data quality dimensions.

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

Definition 3. The *quality* of a dataset D for a task T, is

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

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

This characterization of data quality is more informative since it offers a better understanding of the effect of the differentiations in the data for the task. Under this definition, an effective assessment of the quality of a dataset for a specific task requires both the identification of the data characteristics that are creating large differentiation in the task results and the corresponding computation of the sensitivity factor for them. The challenging task in this process is the specification of that factor. Figure 3.1 illustrates the overall theoretical framework for the contextual data quality problem.

3.4 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 results of the task. To do this, we have developed a framework that tests in a systematic way the effect that a change in the dataset

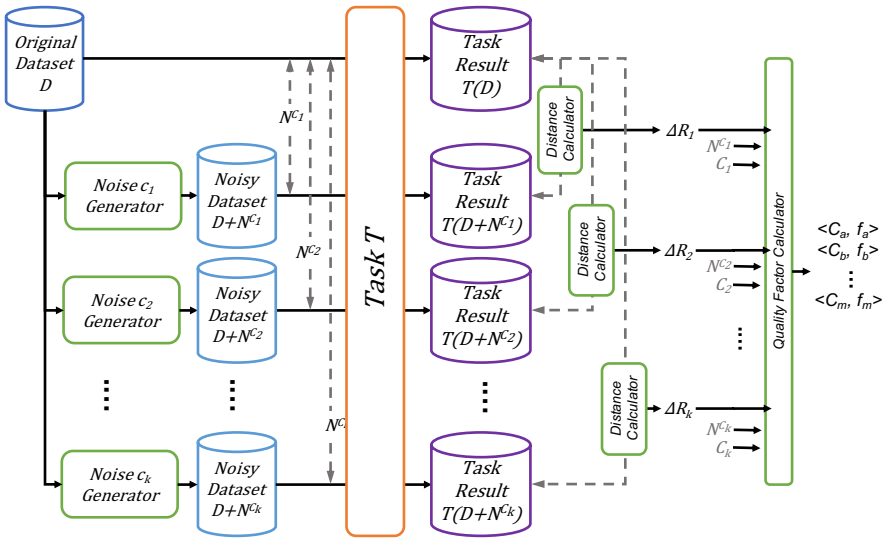

Figure 3.2: A Data Quality Evaluation Framework

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 c. We refer to the changes as *noise* and the characteristic c that they are affecting as the *type* of the noise. For every test performed, the system builds a *scenario*: a triple $\langle c, N^c, \Delta R \rangle$, where c is a characteristic, N^c is some amount of noise of type c, and $\Delta R = d(T(D), T(D + N^c))$ is the change in the results of T when applied to D with and without the noise N^c.

Figure 3.2 depicts the process performed by the framework. 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 computes the degree of the impact on the results of each noise type. The computed degree constitutes the sensitivity factor of the task for that specific type of noise. Note that the process does not assume the input dataset to be clean. Our idea is that the application of this framework still generates an estimation of the errors that are mostly affecting the results of the task. If the dataset is completely broken, the results generated by our framework are not useful. For this reason, we stress the point that this system is not thought to be a replacement for current data profiling and quality evaluation systems, but the two applications should work together to

benefit one from the other. In fact, in the previous example the application of data profiling frameworks would have already suggested to the user the need for repairing the data.

Algorithm 1 illustrates the process in details. The framework takes as input a dataset D, a task T to apply, a set of noise types N 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. For example, the calculation of the maximum value of a column on the original dataset gives a result. If a noise generator affects that value, then the result would be different for that noisy instance. Hence, multiple runs enable a more objective analysis and allow the user to understand the real effects of the noise.

First, we apply the task T on the original dataset D (line 2), and the results will be used along all the process. Our experiments identified 4 types of behaviors (see Section 3.8) that represent how the results change increasing the 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 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 3.7, 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 takes the minimum and the maximum noise percentage values of P. The system then generates the 2 noisy instances (line 20) over which the given task is run (line 21). Then, the framework measures the distance of the results of each of the noisy instances from those of the original dataset (line 22). If their distance is lower than the given threshold (line 9), it suggests that the relationship is either constant or parabolic (line 16). To discriminate the two,

Algorithm 1 ApplyF4U

Require: Dataset D, task T, set of noises N, ordered set of percentages P, threshold ϵ, number of cycles k
Ensure: *sensitivityFactors*, one quality factor for each noise in N

```
1:  sensitivityFactors = {}
2:  T(D) ← runTask(T, D)
3:  for all c ∈ N do
4:      factors = {}
5:      while k > 0 do
6:          S ← {} ; k ← k − 1
7:          S.add(computeScenario(T, D, c, P[0], T(D)))
8:          S.add(computeScenario(T, D, c, P[P.length − 1], T(D)))
9:          if |S[0].ΔR − S[1].ΔR| < ε then
10:             S.add(computeScenario(T, D, c, P[P.length/2], T(D)))
11:         else
12:             for all i ∈ 1 ... (P.length − 2) do
13:                 S.add(computeScenario(T, D, c, P[i], T(D)))
14:         factors.add(measureSensitivity(S))          ▷ Algorithm 3
15:     noiseFactor ← avg(factors)
16:     behavior ← determineBehavior(noiseFactor)
17:     sensitivityFactors.add(⟨c, noiseFactor, behavior⟩)
18: return sensitivityFactors

19: function computeScenario(T, D, noise, p, T(D))
20:     D_n ← generateNoise(noise, D, p)                ▷ Section 3.5
21:     T(D_n) ← runTask(T, D_n)
22:     ΔR ← measureDistance(T(D), T(D_n))              ▷ Algorithm 2
23:     return ⟨noise, p, ΔR⟩
```

we build a scenario for the noise percentages in the middle of the range in P (line 10).

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 (line 14). Once we repeated the process k times, we compute the average of the sensitivity factors (line 15), which can be implemented in multiple ways (Section 3.7). The AVG function enables the classification of the pattern the sensitivity factor follows for the current noise (line 16). Finally, we return the set of the sensitivity factors computed, one per noise (line 18).

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

3.5 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. 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 [ATV08], as well as many practical scenarios. Following an approach similar to BART [Aro+15], 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, 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 creation of custom generators, as other data-generation methods do [Rat+17], but comes also with a list of pre-developed such components.

[Null] This generator makes parts of the dataset unknown by converting them to *null* values.

[Missing Info] This generator removes parts of the dataset to increase its incompleteness. It does so by deleting a portion of rows of the dataset specified as an input parameter.

[Edit] The generator performs insertions, deletions, and substitutions on parts of the values that exist in the dataset.

[Permutation] Given a value, the permutation generator scans the content and reproduces a number of permutations of consecutive symbols.

[Abbreviation] This generator takes some data values from the dataset and turns them into abbreviations by maintaining only the first few symbols.

[FD Violation] Given a functional dependency, it scans the dataset for records with the same values in the attributes specified in the head of

some functional dependency and modifies the values of the attributes specified in its body, to ensure the violation of the dependency.

[Shuffling] This generator applies only to strings and chooses randomly two consecutive words and swaps them.

[Acronym] It applies to strings and replaces values with multiple words, with the initials of these words.

[Synonym] It selects a word of an attribute value in the dataset and replaces it with a synonym.

[Multilingual] Similarly to the synonym generator, the multilingual substitutes a word with its translation in a given different language.

[Base Change] The goal of the current generator is to change the base of the numbers, e.g., turning a decimal number to its binary or hexadecimal representation.

[Scale] The generator intends to scale up/down values by multiplying/dividing them by a given factor.

[Negation] It negates a portion of the values in the dataset by turning them from positive to negative or vice versa.

[Modification] This generator adds or subtracts some fixed value to numbers in the dataset.

Composite generator

The previously mentioned generators introduce in the data a specific kind of noise. However, real-world data has a mix of different types of noise. The composite generator is a meta-generator that introduces mixed noise by iteratively calling different generators, with the output of one call that becomes the input of the next generator.

3.6 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.

Algorithm 2 MEASUREDISTANCE

Require: X, Y: two relational tables
Ensure: *distance* between the tables X and Y

```
1:  distances ← {}
2:  for all s ∈ X do
3:      for all t ∈ Y do
4:          sim ← 0
5:          for all att ∈ X.attributes do
6:              s_shingles ← CREATESHINGLES(s.att)
7:              t_shingles ← CREATESHINGLES(t.att)
8:              sim ← sim + SIMILARITY(s_shingles, t_shingles)
9:          distances.add(⟨(s, t), 1 - (sim / |s.attributes|)⟩)
10: assignments ← MATCHING(distances)                    ▷ Section 3.6.3
11: distance ← 0
12: for all assignment ∈ assignments do
13:     distance ← distance+assignment.distance
14: return distance
```

3.6.1 Task-specific Metrics

For many data analytic tasks, there are well-established metrics for quantifying their effectiveness. As an indication, for clustering can be used the Fowlkes-Mallows score [FM83], the Silhouette coefficient, or the Rand Index [VEB10], the F1 score, the accuracy, the precision or recall for classification [Pow11], and the Mean Squared Log Error, or the R^2 score for regression [WM05]. In these cases, the difference of their scores before and after the introduction of the noise in a dataset gives an indication of the impact of the noise on the task results. It is true that the changes in the results depend on the application of the noise on the key features used by the task but the system generates the noise and measure the difference multiple times for minimizing the randomness and normalize the results.

3.6.2 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. Appendix A explains some techniques we identified, i.e., nulls, entropy, and value cardinality.

3.6.3 Universal Distance Metric

In the very general case of some complex analytic task for which there is no established evaluation metric, in order to measure the difference between two task results, it is possible to perform a direct comparison of their contents. This kind of comparison is the most generic, and can always be computed, even when there are already established metrics for the task. The challenge of the approach, however, is the performance.

To compute the distance between the contents of two task results, we consider each result set as a distinct set of tuples, and we identify their best possible bipartite matching. A tuple in one result set is matched to a tuple in the other result set that is highly similar (if not identical). The best bipartite matching is the one that minimizes the sum of the distances among the elements of the two result sets. The algorithm requires that the cardinality of the two datasets is equal, so we introduce in the result set that has the fewer elements some special "dummy" tuples, which have the maximum distance from any other tuple. Algorithm 2 describes the computation. First, we compute the distance between every pair of tuples, one from each of the two result sets. We use shingles to improve the robustness to the changes in the data (line 6-7). The distance between the attribute value of two tuples is computed with the Jaccard distance of their respective sets of shingles (line 8). We normalize the sum of the attribute distances (line 9), and, at the end of this process, we have a bipartite graph, with the elements of the two datasets as nodes and the distance between the tuples of the two datasets as the edges. The MinHash and LSH techniques are exploited to avoid computing the distances across all the possible pair of tuples.

We provide two alternative methods (a greedy and an optimal) for implementing the matching strategy that we describe below. Once the desired matching has been identified, we calculate the sum of the distances of the selected edges that is the distance between the two result sets. Note that this distance is a metric, i.e., the distance between two elements is never negative, the distance between an element and itself is 0, and the triangular inequality is satisfied.

The Greedy algorithm for the bipartite matching, having marked all the nodes (i.e., tuples) and the edges as unselected, ranks the edges

according to the distance of the two tuples it connects. From the unselected edges, it selects the edge with the lowest distance that has both the tuples it connects unselected and marks the edge and the two connected tuples as selected. The algorithm repeats the process until no more edges can be marked as selected. The set of the selected edges is the generated matching. Given n tuples, the complexity of this approach is $\mathcal{O}(n^2)$, since it first ranks the edges ($\mathcal{O}(n \log n)$), and then it performs a nested cycle over the elements ($\mathcal{O}(n^2)$) to identify the best matches. Note that this greedy approach provides an approximate solution.

Instead of the Greedy, any other bipartite matching algorithm can be used. For instance, we used the Hungarian algorithm for the optimal solution but, as expected, it is slower than the Greedy ($\mathcal{O}(n^3)$), and requires more space ($\mathcal{O}(n^2)$).

3.7 SENSITIVITY FACTOR COMPUTATION

We first introduced different types and amounts of noise in the original dataset, and second, we measured the difference between the result sets of the task of interest on the original and the artificially created noisy datasets. The last step is the computation of a score that indicates the relation between the noise and its impact on the analytic results. This score is referred to as the *sensitivity factor* (Definition 3) and is detailed in Algorithm 3. We assume as input a series of scenarios, i.e., items of the form $\langle c, N^c, \Delta R \rangle$, related to a single noise type c. Then, we compute a sensitivity factor for the noise c by considering the set of $\langle N^c, \Delta R \rangle$ 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 [Net+96], which identifies a linear relation between the two variables (line 6). The second method we employ is the polynomial regression, capturing when the relation is polynomial (line 7). 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

Algorithm 3 MEASURESENSITIVITY

Require: S: a set of scenarios for a data characteristic c
Ensure: $\langle(r_{score}, r_{slope}), (p_{score}, p_{slope}), \rho\rangle$, computed between the noise amounts and the result distances contained in S, where
(r_{score}, r_{slope}) is the linear regression score and slope,
(p_{score}, p_{slope}) is the polynomial regression score and slope,
ρ is the Spearman correlation coefficient

```
1: noises ← {}
2: distances ← {}
3: for all scenario ∈ S do
4:     noises.append(scenario.N^c)
5:     distances.append(scenario.ΔR)
6: (r_score, r_slope) ← LinearRegression(noises, distances)
7: (p_score, p_slope) ← PolynomialRegression(noises, distances)
8: ρ ← SpearmanCoefficient(noises, distances)
9: return ⟨(r_score, r_slope), (p_score, p_slope), ρ⟩
```

1 (perfect fit). The third method is the Spearman correlation [Spe04] (line 8) 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 3.4, we classify these 3 sensitivity factors in 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. Finally, the irregular pattern shows a low linear and polynomial score and a higher standard deviation.

3.8 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

Name	Rows	Numeric Feat.	Textual Feat.
ADULT	48842	6	8
BANK	45211	7	9
AIRLINES	4121943	20	6
TIM (10% sample)	416304	108	62

Table 3.2: Datasets Characteristics.

noise, its amount, the dataset at hand, 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 ability of the proposed generic distance function to provide valuable insights in the absence of a task-specific evaluation method; and (iv) the scalability of our system.

System Description: The experiments were performed on a cluster of 6 machines, each with 16 cores and 96 GB RAM running `Ubuntu 14.04 LTS`. Each experiment run with a parameter `k` (described in Section 3.4) equals to 5, and the plots report the mean across the runs. The tasks considered are clustering, classification, and regression, and the solutions implemented are k-Means, Random Forest [Bre01], and Linear Least Square, respectively. The noise generators, the generic distance measure, and the tasks are written in Python 3.6 on Apache Spark 2.2 and with Scikit-learn. We run 10 noise generators with 10 different percentages (5, 10, 20 … 90%). Source code: `https://github.com/forons/contextual-dataquality`.

Datasets: We tested our framework on 4 different datasets, summarized in Table 3.2, this allows us to obtain results from different domains and use cases.

(i) `ADULT` contains demographic information about US citizens, such as age, work, annual income, hours worked per week, and marital status [LN16]. In the literature, all these features have been used to predict whether a person earns more or less than 50K $ per year, and we used the same classes to configure the classification and clustering tasks. In regression, instead, we predicted the number of hours worked per week, considering only the numerical features.

(ii) BANK records information about mobile marketing campaigns of a banking institution. It contains clients age, balance, housing, job, marital status, default state, and personal loan; as well as info about the campaign (contact type, contact month and day of the week, and contact duration) [MLC11]. The attributes were used to predict whether a client would subscribe or not to a term deposit, and to similarly detect the two clusters of customers. Finally, in regression, we used the numerical features to predict the user account balance.

(iii) AIRLINES describes domestic flights collected from the US Department of Transportation and includes the itinerary fare, number of passengers, starting and arriving airport, round-trip indicator, and miles flown [Par+18]. Also, we obtained 4 classes of flights by dividing the ticket cost into 4 quartiles, i.e., cheap, medium, high, and expensive. We used these classes in both the classification and clustering task, while in regression, we predicted the ticket cost using all the numerical features. For the task evaluation, we used a sample of 10000 rows, while for testing the scalability of the system, we used incremental portions of this dataset.

(iv) Finally, we used a 10% random sample of the dataset provided by TIM, the largest Italian telco, that contains the (technical only) information related to trouble tickets opened by landline (fibre) users and their subsequent management, such as device information, location, problem, and intervention data. Details about how we use the features are presented in Section 3.8.5.

3.8.1 Illustrative Use Case

To better understand how our framework works, we now consider the AIRLINES dataset and the classification task to predict the price range of a flight using all the other attributes. The results of a contextual data quality assessment should answer the following: (**Q1**) which errors compromise more the quality of the results? and (**Q2.**) can we ignore some errors without compromising the quality of the results?

Note that we assume neither that the dataset is clean nor any specific prior knowledge of its characteristics. We emphasize that even in case of a dirty dataset, the framework will still gauge the trends.

Moreover, this framework is thought to be a help to the user, but also to be used together with standard data profiling tools, to grasp the datasets that are completely broken.

This system first 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 comparing the F1 score. Figure 3.4 (middle plot) reports the results of the analysis provided by the framework. The closer is the F1 on a noisy instance 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 are those with the largest impact on the results. The motivation behind this behavior is that these noises produce uncertainty in the classification model by removing useful information from the training dataset. 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, the time spent by an analyst to clean or repair these errors would not bring any significant benefit to the results. Upon closer inspection, we can justify the behavior of `SHUFFLING` by the absence of fields with multiple words, which means that the `SHUFFLING` error does not introduce any change. Even more interestingly, `EDIT` errors in both textual and numerical features impact the results much more than the errors introduced in only numerical features, as in the case of `BASE CHANGE` and `NEGATION`.

These observations are confirmed in Table 3.3 (refer to the third row and second column), where it presents the sensitivity factor computed by the framework for each noise. This value estimates the effect of the noise on the results of the task exploiting the relationship between the F1 score and the percentage of noise introduced in the data. For each group, the first column reports the linear dependency between these two variables, the second provides their polynomial relationship, and the third is their Spearman correlation (the symbol '*' identifies statistically significant elements with p-value < 0.05). We can see that there exists a linear correlation (L in the *Class* column) between the degradation of the quality of the task results and both

the NULL and the MISSING INFO errors with coefficient 0.96 and 0.99 respectively. The latter, however, presents a leaning slope (-0.81 vs. -0.97) meaning that MISSING INFO has a higher impact than NULL.

The BASE CHANGE and NEGATION noises have a very low linear coefficient (first column, first value in the table), whereas their polynomial coefficients are very high, meaning that there is a polynomial relationship (denoted by P in the *Class* column). That is, changing the sign of all the values in the dataset does not affect the results of the classification task. The values of the *Linear* column also point out a linear impact due to the noise introduced by PERMUTATION and EDIT. Conversely, the Spearman correlation and the plot signal a constant behavior for SHUFFLING and ACRONYM, i.e., these errors have no effect on the task for the current dataset.

To sum up, referring to our initial questions, the system suggests that MISSING INFO and NULL are the most problematic errors for classification followed by PERMUTATION and EDIT (**Q1**), while SHUFFLING and ACRONYM can be safely ignored (**Q2**). Moreover, the framework shows that typos in textual features are more troubling than in numerical features, which is quite surprising given that the dataset contains more numerical features than textual features.

The above shows the ability of our framework to provide valuable insights into the relationship between error types and data mining tasks, without the need to be familiar with the dataset at hand, which is a common situation in the case of exploratory analytics [IPC15; Mot+17]. These insights allow the analysts to focus their attention on fixing only the most problematic errors in the dataset. Moreover, a common procedure is the auditing of a sample of the dataset to gauge the presence of data quality issues in the complete dataset [Sch+18; Bat+15]. Our framework can help the analyst also in these cases. Assume that an auditing suggests that the dataset has 30% of MISSING INFO and 90% of NEGATION issues, due, for example, to the wrong parsing of the hyphen symbol. Then, the results above clearly prompts the analyst to prioritize the cleaning of the missing tuples, since 30% of MISSING INFO affect the results more than 90% of NEGATION.

3.8.2 Understanding the changes in the results

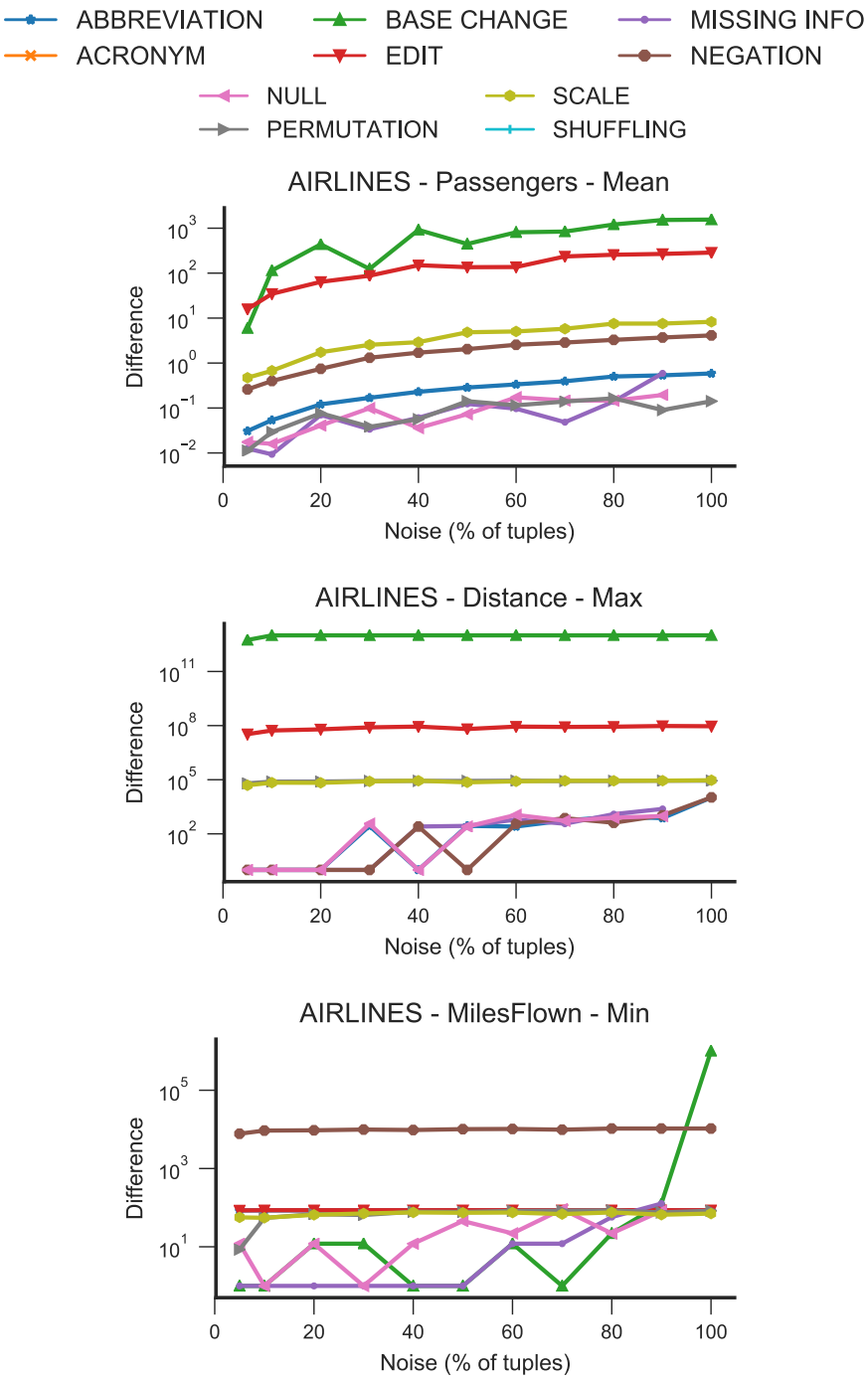

Figure 3.3: `AIRLINES` aggregations on `Passengers` column (mean), `Distance` (max), `MilesFlown` (min), from top to bottom.

In the following, we characterize the variations in the results for different datasets and tasks.

Profiling Aggregation Validation

We validate the ability of our framework to detect the problematic data quality issues for 3 simple data profiling aggregations [Nau13], i.e., finding the mean, maximum, and minimum value in a given column of the dataset. These values can be computed exactly and efficiently in the original dataset, hence allowing an easy and accurate comparison with the results in the noisy instances. Figure 3.3 reports the distance of the results in the noisy instances from the ground-truth. The figure at the top compares the mean values.

`BASE CHANGE` and `EDIT` affect the output the most (with a variation up to 10^6 from the mean obtained with the original dataset), since they change the values that will be aggregated in the data profiling task. On the other hand, the behavior of `MISSING INFO` and `NULL` is almost constant, as removing some values does not affect considerably the mean. A similar behavior can be seen in the middle and bottom plots of Figure 3.3, which report the results of the maximum and the minimum aggregation respectively. Note that the cause of the oscillations in the lines is that each noisy instance is generated independently from each other. Appendix B presents an analysis of the other datasets.

To conclude, our framework suggests that fixing the missing values for these analytical tasks is not crucial, while the presence of outliers has a higher impact on the results. These tasks are relatively simple and well understood, and our framework produces results that are compatible with our understanding, hence validating our approach in this baseline scenarios. Our framework can perform the same kind of analysis even for more complex tasks where this kind of insights are not obvious, as presented in the following.

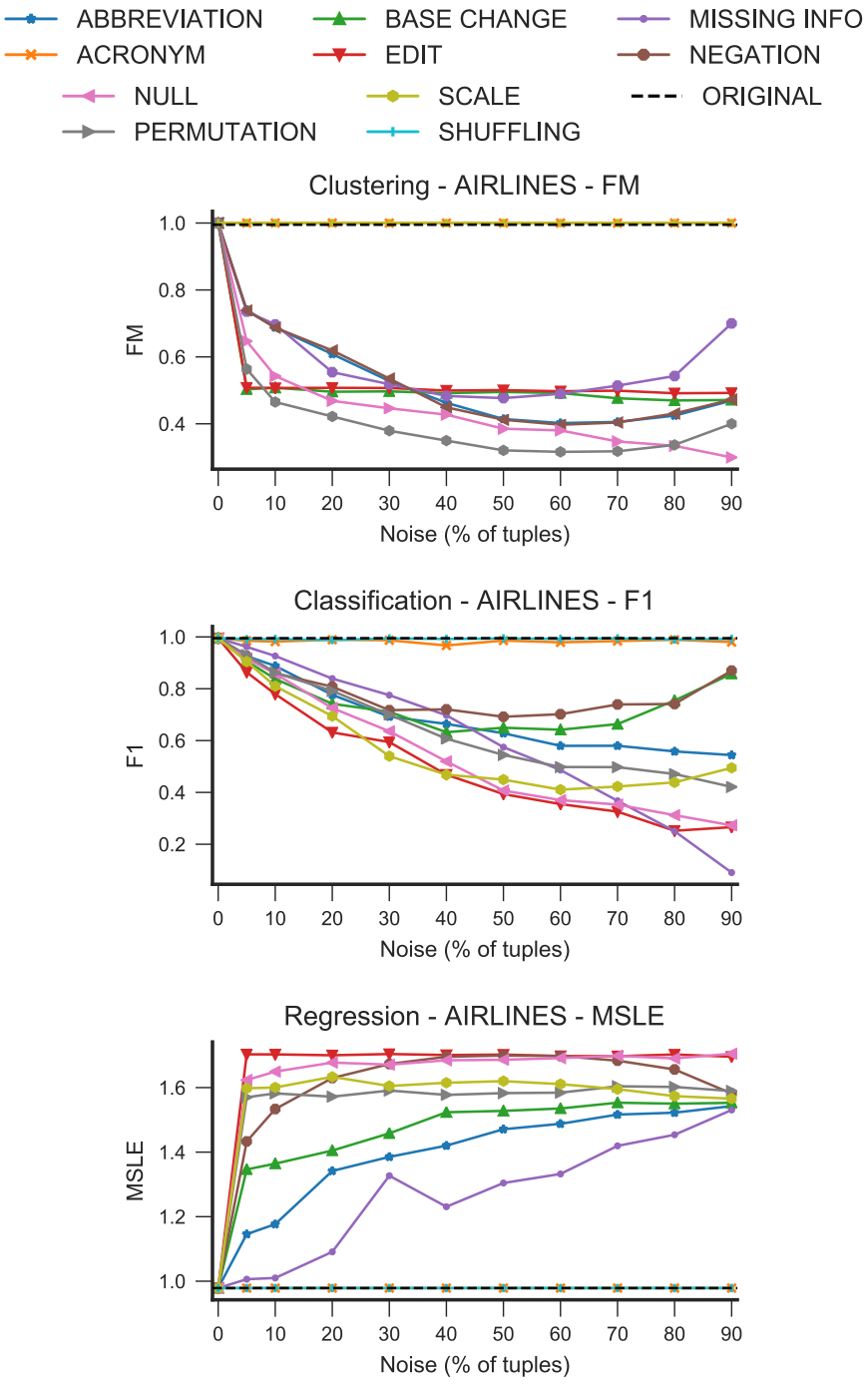

Figure 3.4: Task specific distance for `AIRLINES`, for clustering (top - FM score - higher is better), classification (middle - F1 score - higher is better), and regression (bottom - MSLE - lower is better).

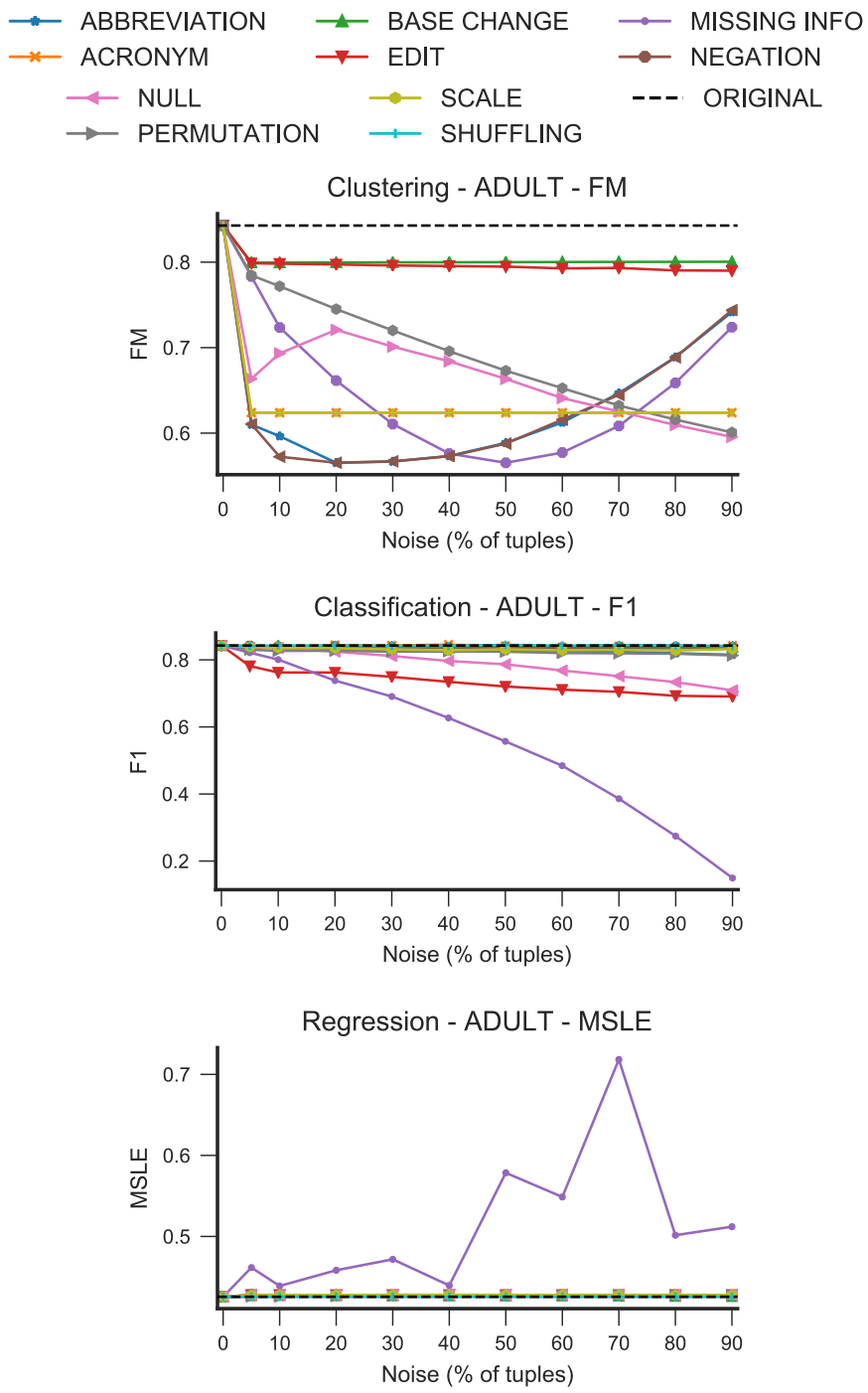

Figure 3.5: Task specific distance for `ADULT`, for clustering (top - FM score - higher is better), classification (middle - F1 score - higher is better), and regression (bottom - MSLE - lower is better).

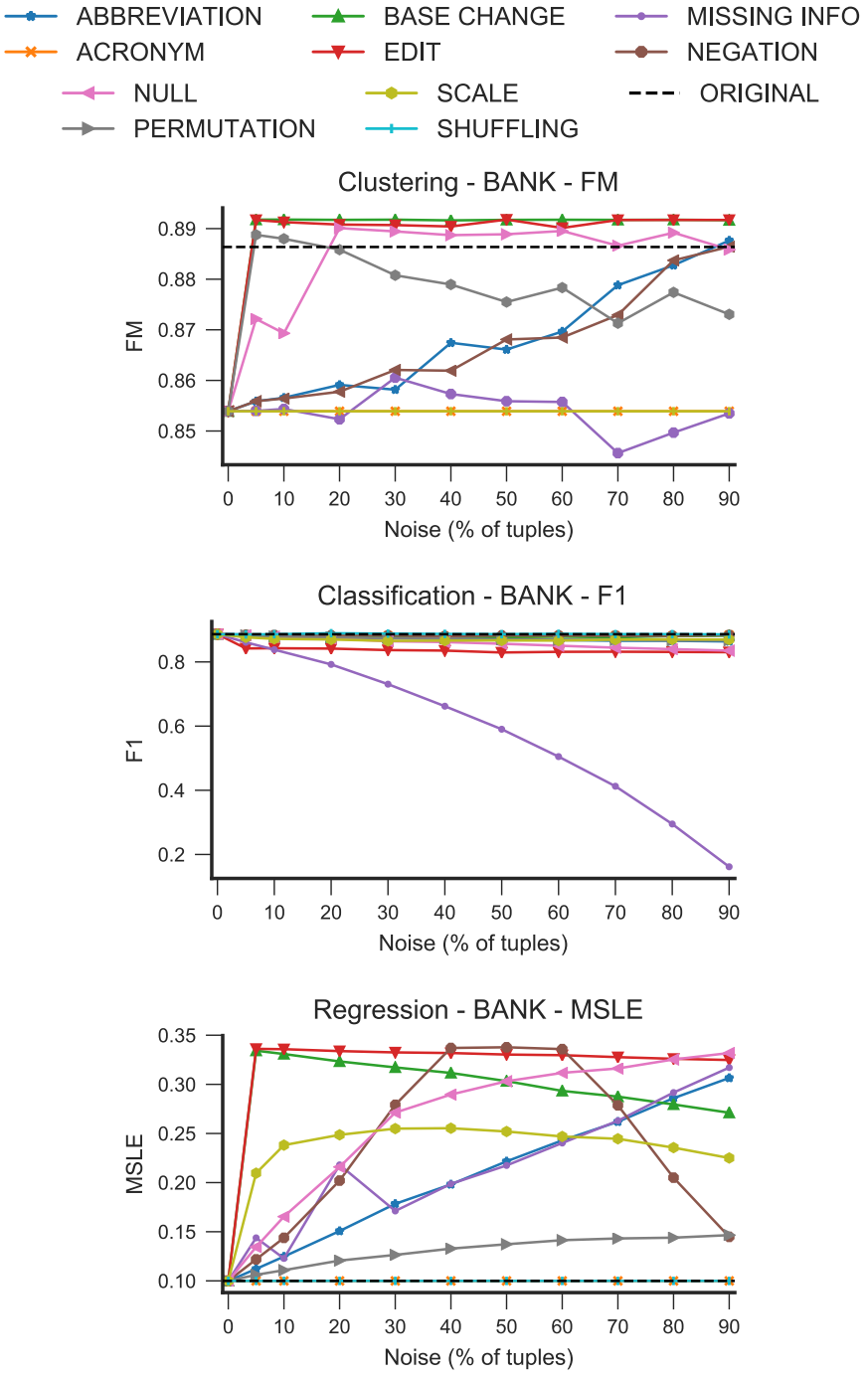

Figure 3.6: Task specific distance for BANK, for clustering (top - FM score - higher is better), classification (middle - F1 score - higher is better), and regression (bottom - MSLE - lower is better).

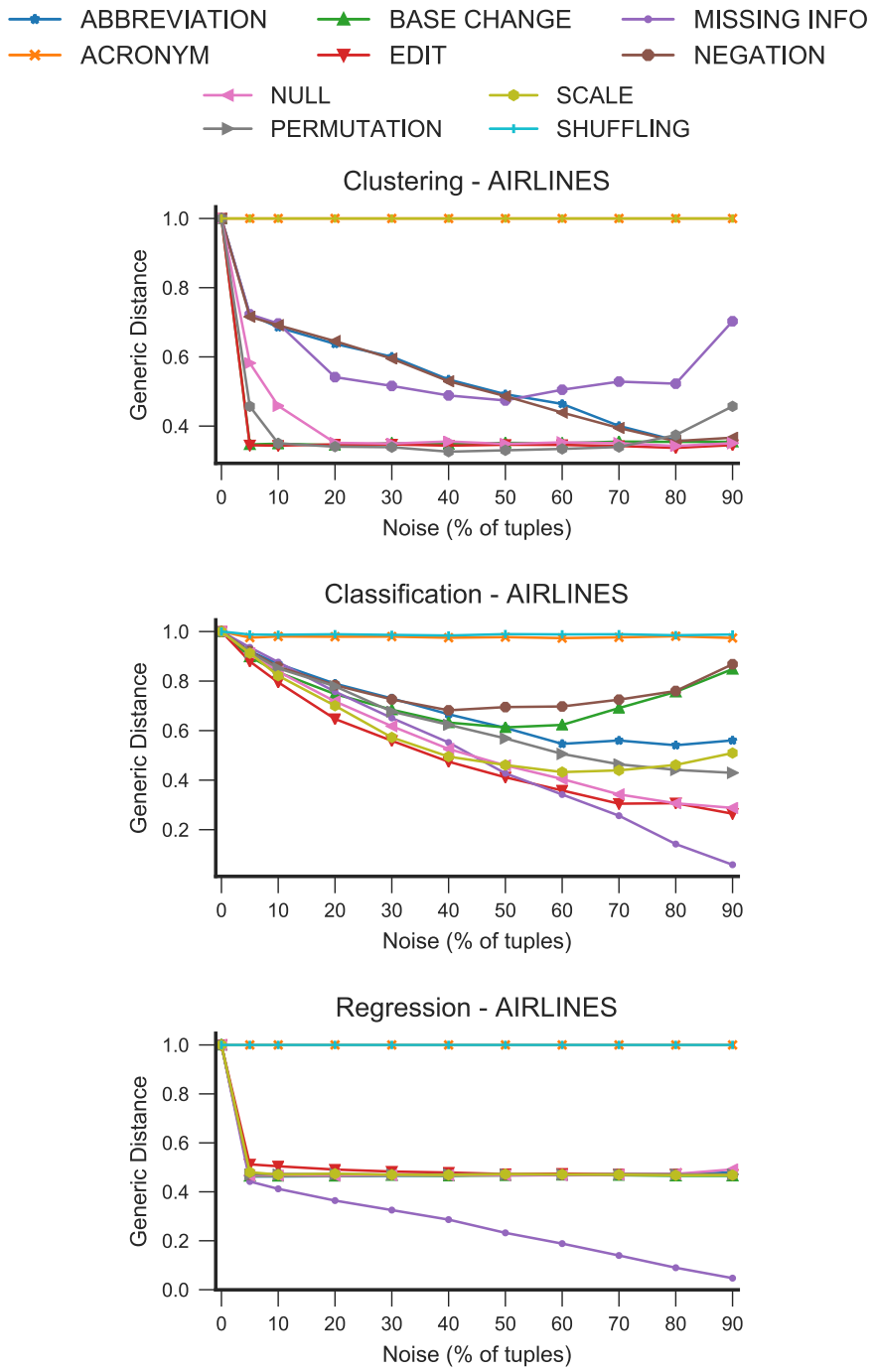

Figure 3.7: Generic distance (higher is better) for clustering (top), classification (middle) and regression (bottom) with `AIRLINES` dataset.

Analysis of complex tasks

We tested the framework with more complex tasks, and in particular, we studied data mining tasks that involve both unsupervised (i.e., clustering) and supervised learning (i.e., classification and regression). We run clustering, classification, and regression on the datasets ADULT, BANK, and AIRLINES. Figures 3.4, 3.5, and 3.6 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 [FM83], which is the mean between precision and recall; 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 2 for the ADULT and BANK datasets, and 4 for AIRLINES.

The clustering plots show once more that the quality of the results depends not only on the amount of noise introduced but also on the dataset itself since the same amount of noise has a different impact in different datasets. For instance, the BANK dataset is less affected by the presence of noise than the other datasets. The effect of NEGATION is much more prominent in the ADULT dataset than in the AIRLINES. In both datasets, NEGATION follows a parabolic behavior, but around 50% it is the most impacting noise for ADULT, leading to severe damages in the results.

Classification (middle row) has in general worse results in ADULT than in BANK (lowest value around 0.7 vs. 0.8 without considering MISSING INFO). On the other hand, AIRLINES presents the most interesting results, as NULL, EDIT, and MISSING INFO errors affect the results surprisingly in a very similar way (at 80% of noise they all present an F1 score around 0.3, which is very close to the results obtained with a baseline model). In the regression task, AIRLINES is the most noise-sensitive dataset, while ADULT is the least affected, except for MISSING that has a significant effect on the quality of the predictor (MSLE of 0.7 with 70% of noise).

To conclude, the experiments substantiate our claim that knowing that a dataset is affected by some data quality issues is not enough to have a context around the dataset. Even in the case of complex tasks,

our framework can assist the analyst in uncovering how the results are affected by the data quality issue, hence providing the necessary context to define a strategy to clean the dataset most effectively and efficiently.

The effect of noise on different tasks

We now focus on the AIRLINES dataset (Figure 3.4). For this 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

In the previous sections, we proved that the quality of the results of a task depends on the type of noise in the dataset, its amount, the task, and the dataset themselves. To quantify the effect of the noise more precisely, our framework computes a set of *sensitivity factors* for each task, dataset, and noise type, which measure the effect of the noise on the results of that task and that dataset. Table 3.3 reports all

Task Distance		Clustering			Classification				Regression			
Noise		**Linear**	**Polynomial**	ρ	**Class**	**Linear**	**Polynomial**	ρ	**Class**	**Linear**	**Polynomial**	ρ	**Class**
ADULT	ABBREVIATION	(0.00, -0.01)	(0.21, -0.41)	**0.13**	I	(0.72, -0.02)	(0.86, -0.05)	−0.81*	C	(0.15, 0.00)	(0.74, 0.01)	0.56	C
	ACRONYM	(0.18, -0.08)	(0.37, -0.36)	**−0.48**	I	(0.02, -0.00)	(0.03, -0.00)	−0.28	C	(1.00, 0.00)	(1.00, 0.00)	0.00	C
	BASE	(0.16, -0.01)	(0.35, -0.07)	0.54	C	(0.12, -0.00)	(0.61, -0.03)	−0.41	C	(0.26, -0.00)	(0.49, 0.00)	−0.52	C
	CHANGE												
	EDIT	(0.37, -0.03)	(0.51, -0.08)	−0.99*	C	**(0.85, -0.13)**	(0.91, -0.24)	−1.00*	L	(0.16, 0.00)	(0.33, 0.00)	−0.05	C
	MISSING INFO	–	–	–	–	**(0.97, -0.80)**	(1.00, -0.27)	−1.00*	L	(0.31, 0.13)	(0.42, 0.42)	**0.73***	I
	NEGATION	(0.02, -0.04)	**(1.00, -1.08)**	−0.16	P	(0.06, -0.00)	(0.76, -0.02)	−0.31	C	(0.00, 0.00)	(0.75, 0.01)	0.21	C
	NULL	(0.11, 0.09)	(0.68, -0.67)	**0.41**	I	**(0.95, -0.16)**	(0.99, -0.04)	−1.00*	L	(0.20, 0.00)	(0.44, 0.00)	0.40	C
	PERMUTATION	(0.70, -0.17)	(0.70, -0.22)	−0.91*	C	(0.78, -0.02)	(0.81, -0.03)	−0.98*	C	(0.86, 0.00)	(0.97, 0.00)	0.99*	C
	SCALE	**(0.96, -0.23)**	(0.99, -0.37)	−1.00*	L	(0.35, -0.01)	(0.92, -0.04)	−0.54	C	(0.16, 0.00)	(0.39, 0.00)	0.00	C
	SHUFFLING	(0.18, -0.08)	(0.37, -0.36)	**−0.48**	I	(0.12, 0.00)	(0.12, 0.00)	0.28	C	(1.00, 0.00)	(1.00, 0.00)	0.00	C
BANK	ABBREVIATION	(0.11, -0.06)	(0.42, 0.29)	**0.52**	I	(0.93, -0.02)	(1.00, -0.05)	−0.99*	C	(0.37, 0.13)	(0.64, 0.54)	**0.54**	I
	ACRONYM	(1.00, 0.00)	(1.00, 0.00)	0.00	C	(0.00, -0.00)	(0.77, -0.03)	−0.07	C	(1.00, 0.00)	(1.00, 0.00)	0.00	C
	BASE	(0.18, 0.01)	(0.36, 0.06)	−0.22	C	(0.04, 0.00)	(0.83, -0.03)	0.18	C	(0.00, 0.01)	(0.24, 0.33)	**−0.54**	I
	CHANGE												
	EDIT	(0.19, 0.01)	(0.36, 0.06)	0.48	C	(0.44, -0.03)	(0.63, -0.10)	−0.88*	C	(0.14, 0.07)	(0.33, 0.38)	**−0.54**	I
	MISSING INFO	–	–	–	–	**(0.96, -0.84)**	(1.00, -0.27)	−1.00*	L	(0.28, 0.33)	(0.31, 0.73)	0.91*	C
	NEGATION	(0.10, -0.00)	(0.12, 0.00)	−0.31	C	(0.04, -0.00)	(0.81, -0.02)	−0.26	C	(0.01, 0.03)	**(0.95, 0.99)**	0.19	P
	NULL	(0.94, 0.04)	(0.98, 0.01)	0.99*	C	(0.99, -0.06)	(1.00, -0.07)	−1.00*	C	**(0.83, 0.22)**	(0.98, 0.56)	1.00*	L
	PERMUTATION	(0.37, 0.02)	(0.78, 0.10)	0.26	C	(0.84, -0.02)	(0.97, -0.04)	−0.97*	C	(0.89, 0.05)	(1.00, 0.10)	0.99*	C
	SCALE	(0.05, -0.01)	(0.16, 0.03)	−0.49	C	(0.09, -0.01)	(0.81, -0.06)	−0.21	C	(0.13, 0.05)	(0.57, 0.34)	0.05	C
	SHUFFLING	(1.00, 0.00)	(1.00, 0.00)	0.00	C	(0.15, -0.00)	(0.59, 0.01)	−0.19	C	(1.00, 0.00)	(1.00, 0.00)	0.00	C
AIRLINES	ABBREVIATION	(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	(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
	CHANGE												
	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

Table 3.3: 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).

the factors computed for the datasets in Figure 3.4, Figure 3.5, and Figure 3.6.

For example, the polynomial slope generated by the NEGATION generator in clustering for the ADULT dataset is very high (-1.08), which means that if the dataset contains even a small amount of negations, cleaning those errors immediately affects the results positively. Similarly, for classification in the BANK dataset, MISSING INFO has a linear coefficient (i.e., 0.96) with an extremely high slope (-0.84), which means that repairing just a few tuples can lead to an improvement in the results. On the other hand, for classification in ADULT, NEGATION 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 Table 3.3 and the results shown in Figure 3.4, Figure 3.5, and Figure 3.6, the data analyst can plan a cleaning process of the dataset at hand. It allows an improvement in the quality of the results of the task she 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.

3.8.3 Generic Distance

We validate the ability of our generic function to measure the distance between the results of a task, by comparing its outcomes with those of the task-specific measures introduced above. Figure 3.7 shows that the generic distance handles very well the case of clustering and classification. Even though the lines do not overlap exactly with those in Figure 3.4, trends, and scales are definitely similar. In AIRLINES the two distances lead to the same conclusions in clustering for ABBREVIATION (polynomial task-specific slope of -1.39 vs. polynomial generic slope of -1.43) and SCALE (polynomial slope with the ad-hoc distance of -1.79 vs. -1.72 with our distance). For the regression task, the generic distance captures only the most prominent behaviors. For example, it is able to highlight the effect of the NEGATION errors (more visible in the AIRLINES datasets, where both are categorized as polynomial patterns). A detailed analysis of the other datasets can be found in Ap-

pendix C. Hence, when ad-hoc measures are not known, our generic distance measure is still able to provide some essential insights.

3.8.4 System Scalability

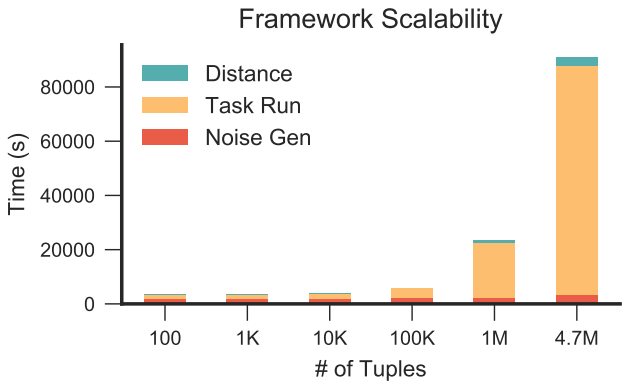

Figure 3.8: `AIRLINES` dataset: System scalability.

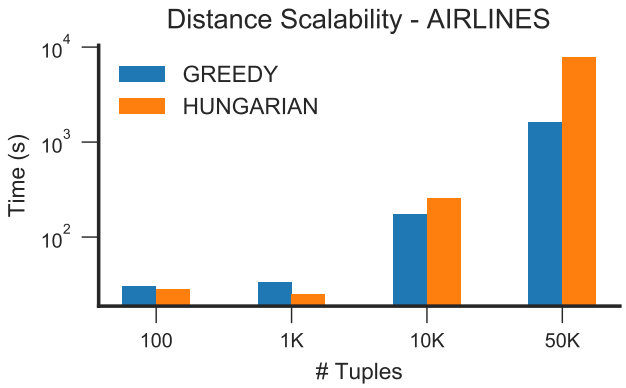

Figure 3.9: `AIRLINES` dataset: Generic distance scalability.

We investigate the scalability of our framework by measuring the running time of the whole cycle, starting from the generation of the noise for each generator to the execution of the task (without the optimizations), and finally to the computation of the distance using a task-specific measure. Figure 3.8 and Figure 3.9 show the performance for the clustering task and the Fowlkes-Mallows score in the `AIRLINES` dataset. In Figure 3.8, we can see that the noise-generation step takes less than 3000 secs to produce 110 different noisy instances

of the dataset (10 noise generators and 11 percentages) even for samples of millions of tuples. A similar running time is required also for the distance measurement (around 3000 secs on the largest dataset for the whole cycle).

The most time-consuming procedure is the execution of the task for each generated noisy instance. Note that the running-time of the task depends on the algorithm chosen, and therefore, the analyst can save time by selecting another algorithm with desirable performances. In addition, the optimization introduced in Section 3.4 can save up to half of the time by creating $\sim 1/2$ of the noisy instances and hence running the task only half times. The analyst can restrict the analysis by sampling the dataset, reducing the noise percentages analyzed, or focusing only on a limited set of noise types. The bottom plot of Figure 3.9 presents the scalability of our custom generic distance equipped with both the Greedy and the Hungarian algorithm. Even though the Hungarian algorithm has worse performance than the Greedy approach, it produces a perfect matching and, consequently, a more accurate distance value.

3.8.5 A Telecommunication Company Application

To provide a real use-case scenario, the data quality framework here introduced has been used in TIM, the main Italian telecommunication company. The company needs to collect information about the trouble tickets opened due to dysfunctions experienced on fiber landlines. Since the total cost of the management of a new ticket is pretty high, including remote and in person interventions, the company needs a way to reduce extra costs introduced by the tickets that are classified as resolved but then are re-opened right after. Thus, they simulated an anomaly detection task running a classification model to predict which tickets are likely to be re-opened after their closure. In addition, we run K-Means to create the cluster of the tickets that will be re-opened and the one about those that will not.

Figure 3.10 presents the results of the experiments conducted over the `TIM` dataset. For clustering, we see that only `NULL` and `MISSING INFO` lead to a low quality of the results. Since we aim at identifying the errors in the data that have higher impact on the results to get

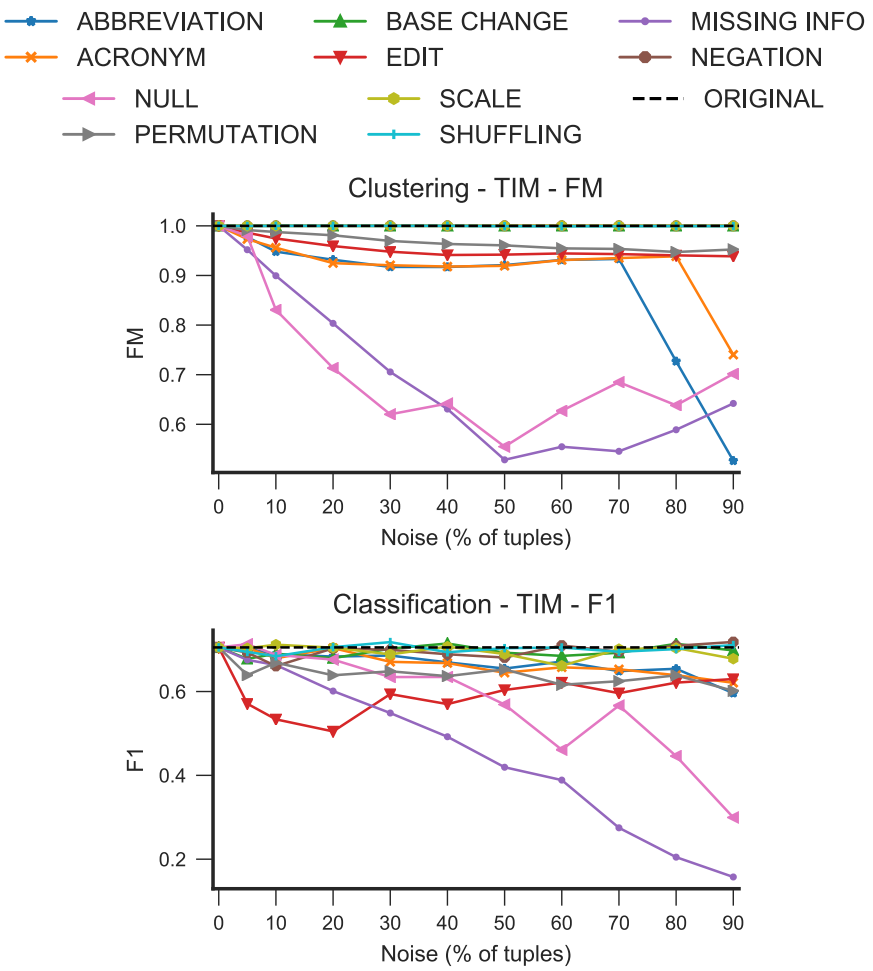

Figure 3.10: `TIM` dataset: Effect on clustering (top) and classification (bottom)

better results, thus, our analysis suggests to clean and repair these two noises primarily, and then focus on the others, starting from `ABBREVIATION` and `SCALE`. For classification, instead, the framework reports worse results, with `MISSING` following a bad pattern, along with `NULL`. In addition, for this task, also `EDIT` errors affect the quality. As a consequence, `ABBREVIATION`, `EDIT`, and `PERMUTATION` have to be considered for cleaning since the impact on the results with respect to those obtained with the original dataset is considerable.

The results above from our framework assisted TIM in identifying the errors present in the data with higher impact on the accuracy score of their anomaly detection algorithm, and to device practices to minimize them in the future.

3.9 SUMMARY

In this chapter, we presented an extension of the notion of data quality that considers the context of the task that is applied on the data. We propose a framework for computing this extended form of data quality by systematically introducing different types and amounts of noise in the dataset and measuring the variation observed in the results of the analytic task. The system then combines the collected observations for computing a sensitivity factor that characterizes the impact of specific data quality issues to the final output of the task. We run extensive experiments over real world and synthetic datasets to access the performance of our approach both quantitatively and qualitatively.

4 MOIRA – CONTEXTUALIZING THE USER GOAL

We move the focus now on the user, which has an important role in any analytical process. We contextualize the user needs in the streaming environment, which emphasizes the differences between the goal that she has in mind. Hence, we aim at dynamically rescheduling the resources of a streaming application enhancing on the user needs, which is, to the best of our knowledge, the first work to do that.

In this chapter, we focus on the optimal solution for the user goal, given the characteristics of the incoming data and the analysis that the user has to perform. Thus, we aim at providing the amount of resources needed to each operator, for satisfying the goal metric defined by the user. The goal metrics that we considered and that the user can optimize are the *throughput* of the deployed job (high number of items processed by the streaming engine), the *latency* (low time required to process an input record), or the *computational cost* (low amount of resources deployed).

4.1 CONTRIBUTIONS AND OUTLINE

The main contributions in this chapter can be summarized as follows:

- To our knowledge, this is the first work that takes into consideration the user goal for an application for a dynamic resource allocation in a streaming environment.
- We provide a formal definition of the optimal solution for the user goal, knowing information about the incoming data, cluster usage metrics, and being aware of the user query. This is also presented in a framework built on top of Apache Flink that enables this kind of analysis.

- We conduct a series of experiments to compare our method with a single static estimation and without any cost estimation, presenting the advantages and the issues for each case.

The remainder of this chapter is organized as follows. We present a use case scenario to allow the readers to understand the needs for such a system in Section 4.2. Then, Section 4.3 exposes the formal definition of the problem we are dealing with, and in Section 4.4 we show the insights of our solution. Section 4.5 presents the experiments performed and the comparison with and without our framework, and with only a cost-based estimation at the deployment of the application.

4.2 MOTIVATING EXAMPLE

A startup has just published a new mobile phone game on the market. They developed a logging system in their mobile application to receive feedback from the users while they are playing their game. The logs have several objectives, such as debugging purpose or reporting, so each kind of log needs a specific management. Moreover, the incoming data to the logging system will change its input rate through time, since people play their game but only for a small amount of time. Even more, if we consider that the game has been purchased more in Europe than in the rest of the world, for sure we will have fluctuating data, with some peaks, for example, in the European evening time and some drops during the European nights. So, in this context, the startup wants to perform some kind of analysis on the logs, to improve the game experience of their customer. This analysis is translated into a DAG (Directed Acyclic Graph), where each node represents an operation of the analysis (e.g., a map/reduce operation), while the edges represent the data flow path. Intuitively, each node will be deployed on a slot placed in a machine of the cluster. A machine has multiple slots available for the nodes of the DAG. However, this DAG is not deployed as it is on the cluster, but it can be optimized in different ways to improve the performance of the analytics. For example, multiple nodes (operations) of the DAG can be performed in the same slot, avoiding the cost of moving the results

of a previous operator to the next operator through the cluster slots that can even be on different machines, or one node can be deployed multiple times to parallelize the operation and thus to improve the performance of that operation.

Focus now on the following analytic described by the startup: get as input the logs, perform a word count of the error code, filter the elements that appear a fewer number of times than a predefined threshold, try to replicate the error with a custom code taken as a black-box, and then save the results into a text file. The analytic, known also as query in the streaming pipelines, will be translated into the DAG pictured in Figure 4.1.

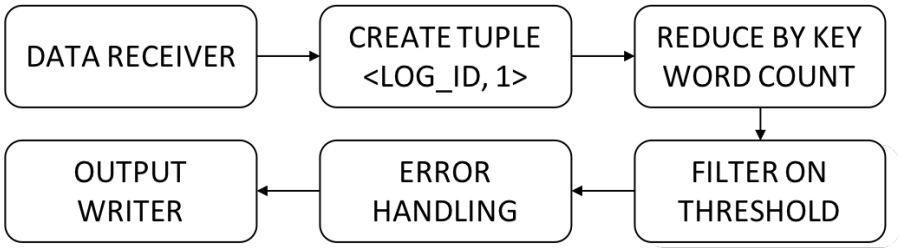

Figure 4.1: DAG example for a use case application

So, with such an analysis, other available systems would optimize the resource allocation or the latency. In our system, it is the user that specifies the optimization goal for the given query. Then, the system will start from it using the gathered information about the incoming data and the cluster usage to deploy the topology that fits better the user's goal. For example, if the user wants to improve latency, one choice can be to chain together the whole process, parallelizing the operators as much as possible. However, if it may be useful while the data is at its peak, when the income rate is on a drop, it becomes not worthy to have the whole cluster filled. Hence, it will be worthy to have a system that takes care of it and decreases the parallelism, which is our aim, even if the goal is to have low latency, since even with fewer resources the goal is still fulfilled.

On the other hand, if the custom error handling policy is a slow operation that needs an amount of resources, for optimizing latency then it will be better to chain the first four nodes and have a bigger parallelism for the error handling task.

Hence, as it is shown, each query has to be handled in different ways given the goal defined by the user. Moreover, since in such a context the data incoming rate might vary through time, a dynamic estimation is needed to improve performance and resource allocation, even better if it follows the user needs and allow the user to fulfill her goal.

4.3 PROBLEM STATEMENT

We assume the existence of a countable set of records $\mathcal{R}$ and a countable ordered set of timestamps $\mathcal{T}$. A sequence of records, each with an assigned timestamp, is referred as a *data stream* or simply a *stream*. Let $\mathcal{S}$ represent the set of all possible streams. We denote as $\tau(r)$ the timestamp of a record r in a stream. The rate of a stream $s \in \mathcal{S}$, at a time t, and for a temporal window w, is the number $\frac{|\{r \mid r \in \mathcal{R} \wedge (t-w) \leqslant \tau(r) \leqslant t\}|}{w}$. Note that the rate of a stream may be different over time.

The records of one or more streams can be processed to produce new objects. There is a number of primitive processing tasks that can be performed on streams. These tasks are referred to as *operators*. The output of an operator is a stream itself.

Definition 4 (Operator). A *stream operator* is a function $o : \mathcal{P}(\mathcal{S}) \to \mathcal{S}$. The set of all possible operators is denoted as $\mathcal{O}$.

Since the output of an operator is a stream, it can be used as input to another operator. In this way, operators can be combined to form more complex processing tasks. Such tasks are referred to as *queries*.

Definition 5 (Query). A k-input query is a tuple $q = \langle \bullet, N, E, I, n_e \rangle$, where $N \subset \mathcal{O}$ and is finite, $\bullet$ is a partial order over N, $n_e \in N$, I is an assignment $[1..k] \to N$ and $E \subseteq N \times N$ such that $\forall \langle n_1, n_2 \rangle \in E : \bullet(n_1) \leqslant \bullet(n_2)$.

A query is actually a function that accepts as input k streams, and produces a single output stream. The output stream is the output of the operator n_e. The assignment I assigns each of the k input streams to one or more operators in N. Intuitively, a query can be seen as a directed acyclic graph that has a node for every operator in N, and

an edge for every entry in E. The node n_e is referred to as the *output* node, and every node that has been assigned at least one input in I, as an *input* node. Input nodes are annotated also with the number of the input stream they have been assigned. For instance, if the assignment I contains the assignment $\langle 3, n \rangle$, it means that the third input stream is among the inputs of the operator n. Input nodes are annotated with the numbers of the inputs that they have been assigned to them. In what follows, when we refer to a query, we will refer to its equivalent graph representation.

Example 1.

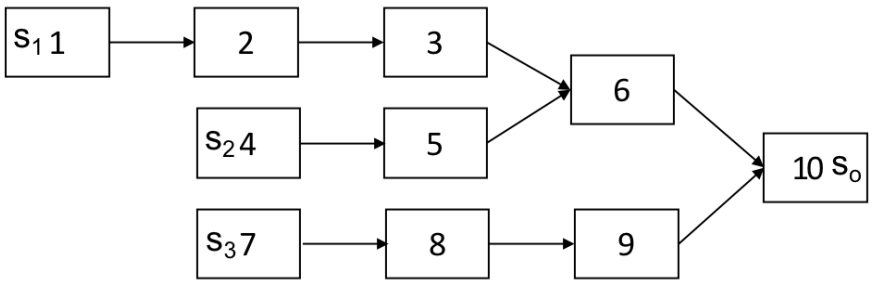

Figure 4.2: Query graph

Figure 4.2 illustrates the graph representation of a query that takes as input 3 input streams. The input nodes are those without incoming edges and their annotations on the left of the identifiers indicate which of the three input streams s_1, s_2, and s_3 they use as input. Although not shown in the graph, note that an input node can have more than one input streams. Node 10 is the output node, i.e., the node of the output stream s_o, which is considered the output of the query.

Since a query is a combination of individual operators, it is possible that different operators are executed on different machines, such that the data produced by one operator is immediately fed to the next operator that can start processing it. Of course, if two consecutive operators are in different machines, then, some cost needs to be paid to transfer data from one to another. If that cost is high, then it may be better to restrict these operators, by bounding them together. This is known as *chain*.

Definition 6 (Chain). Given a query $\langle \bullet, N, E, I, n_e \rangle$, a chain in a sequence of operators $n_1, n_2, \ldots, n_k$, such that for each $i = 1 \ldots k-1$:

- $\langle n_i, n_{i+1} \rangle \in E$
- $\nexists \langle n_i, x \rangle \in E$ such that $x \neq n_{i+1}$,
- $\nexists \langle x, n_{i+1} \rangle \in E$ such that $x \neq n_i$, and
- $\nexists \langle s, n_{i+1} \rangle \in I$

Intuitively, a sequence of two or more operations can form a chain only if the succeeding operator has only one input and it comes from the previous the previous one, apart from the first operator of the chain that can have more than one inputs, and the last element, whose output is not a member of the chain.

Example 2. In the query presented in Example 1, the nodes 7, 8 and 9 can form a chain, since they are consecutive and the only edge that has one endpoint among them is one incoming from 7 and one outgoing to 9. The nodes 3, 6, and 10 cannot form a chain because 6 has an incoming edge that originates from node 5 that is not part of the group.

The nodes of a chain can be collapsed to one node that has as input the input of the first node, and as output the output of the last node. In other words, a chain can be treated as a single operator, and the resulted graph is again a query graph.

The task of an operator can be replicated across different machines such that the different replicas are processing different parts of the records of the input stream. This process is known as parallelization. The parallel versions of the operator are replacing the original operator they parallelize and are called *replicas*. All the replicas of an operator have the same input as the original operator and the same output.

Definition 7 (Parallelization). A k-parallelization of an operator $n \in N$ of a query $\langle \bullet, N, E, I, n_e \rangle$, is a set of replicas $n_1, n_2, \ldots, n_k$ of the op-

erator n, such that a new query can be created of the form $\langle \bullet, N', E', I', n_e \rangle$ for which:

$$
\begin{aligned}
N' &= (N - \{n\}) \cup \{n_1, n_2, \ldots, n_k\}, \\
E' &= \{\langle n', x\rangle \mid n' \in \{n_1, \ldots, n_k\} \wedge \langle n, x\rangle \in E\} \\
&\quad \cup \{\langle x, n'\rangle \mid n' \in \{n_1, \ldots, n_k\} \wedge \langle x, n\rangle \in E\} \\
&\quad \cup \{\langle x, y\rangle \mid \langle x, y\rangle \in E \wedge y \neq n \wedge x \neq n\}, \\
I' &= \{\langle s, x\rangle \mid \langle s, x\rangle \in I \wedge x \neq n\} \\
&\quad \cup \{\langle s, n'\rangle \mid \langle s, n\rangle \in I \wedge n' \in \{n_1, \ldots, n_k\}
\end{aligned}
$$

There are different ways to execute a query depending on what operators are parallelized and what are executed in sequence on the same machine. Each different way of doing this has a different cost. Parallel executions can exploit different cores at the same time, but increase the data communication cost. Execution on the same machine, on the other hand, is increasing the time since the operators are executed in sequence, but saves communication cost. To model the way a query can be executed, we define the notion of an execution plan. Intuitively, the execution plan is a specification of what operators should be chained and what should be parallelized.

Given a query $\langle \bullet, N, E, I, n_e \rangle$, we consider a number of equivalent classes, as many as the number operators, i.e., $|N|$. Each equivalent class is modeled by its representative. By default we assume that every operator belongs to a different equivalent class, which means that each operator is also the representative of the class to which it belongs. Deciding that two or more operators need to be executed on the same machine, can be modeled by simply putting the two operators in the same equivalent class. Merging two equivalent classes is as simple as making the members of the second class have as a representative the one from the first class. This means that we can model the chains by a vector that consists of as many elements as the number of operators in the query, and each element indicates the representative of the equivalent class in which the respective operator belongs.

In a similar fashion we can model the parallelization by indicating the degree of replications that we need to achieve for each operator we need to parallelize. This means that we can also represent the parallelization as a vector of integers, one for each operator. Note

that parallelization is done only for operators that are not chained, which means that for such a vector to be valid, an element can have a value more than 1 only if the respective operator is the only member of its equivalent class (which intuitively translates to not being part of a chain).

The combination of the two vectors, one for the chains and one for the parallelization, is what we refer to as a *execution plan*.

Definition 8 (Execution Plan). Given a query $\langle \bullet, N, E, I, n_e \rangle$, an execution plan is a tuple $\langle C, P \rangle$, where C is a vector of $|N|$ elements, each one with a value from N, and P is a vector of positive integers.

An execution plan $\langle C, P \rangle$ is said to be valid if for every $i=1..|P|$ with $P[i]>1$, it holds that $|C[i]|=1$. By abuse of notation, we use $C[i]$ to denote the equivalent class of the operator i-th operator, and the $|C[i]|$ to denote the cardinality of that equivalent class.

The performance of the execution of a query at any given time depends in the data that arrives into its input stream and the execution plan that has been followed. Our goal is to be able to decide at any given moment, given the input stream data, what the best execution plan is.

[Problem Statement]: Given a query $\langle \bullet, N, E, I, n_e \rangle$, and a series of tuples of the form $\langle S, p, c \rangle$, where S is a set of input streams, p is an execution plan and c is a cost of that plan, we would like to find the best execution plan when the set of input streams is S'.

4.4 MOIRA ARCHITECTURE

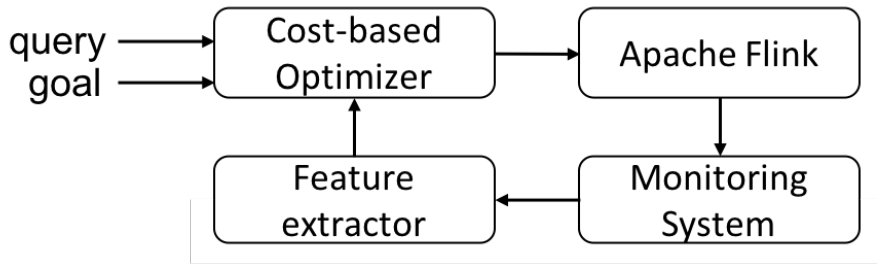

Figure 4.3: Moira framework

In this section, we present Moira, the framework we propose as a solution to the problem of dynamically applying the best execution plan to the user queries. Moira is a system built, for our scope, on top of Apache Flink, but it can be adapted to any other available streaming framework. As shown in Figure 4.3, it is composed of three main components and it strictly interacts with Apache Flink. The entry point of the system is the cost-based optimizer component, which takes as input the user query over the available data and her goal. Note that our framework is goal-centered since each query has to be optimized differently according to the user needs. This component finds the execution plan that fits mostly the given user goal and sends the built application to Apache Flink in order to run it. The streaming framework will then deploy the application through the cluster machine.

Meanwhile, the monitoring system is actually polling the system to gather information about the cluster usage, the characteristics of the data, e.g., the income rate, and the status of the running applications. This information will be then sent to the dynamic cost-estimator that will build the features to use in the cost estimation that will check again if the already deployed topology is the best one or if the newly gathered information will allow a better execution plan to be used. In the following, we present each component in details.

4.4.1 Cost-based Optimizer

The purpose of the cost-based optimizer component is to derive the optimal topology for executing a query based on the available resources and on an optimization goal that was defined together with the query by the Data Stream Processing (DSP) application programmer.

The optimization goal describes how resources should be allocated by the cost-based optimizer. It is defined as a weighted triple of three conflicting optimization areas: cost, latency, and throughput, i.e. $\langle W_{\mathtt{cost}}, W_{\mathtt{latency}}, W_{\mathtt{throughput}} \rangle$. Here the term cost refers to the economical allocation of resources, allowing to execute the query using fewer slots and consequently fewer machines in a cluster. This allows either to limit expenses for compute nodes, e.g. if deployed

in public cloud, or to reserve resources for deploying other topologies in parallel on a cluster with limited resources. The latency goal quantifies the importance of generating output with a small delay after an event is entering the topology through an input stream, while throughput emphasizes a topology's ability to absorb high input rates. The optimization goal gives the DSP application programmer the possibility to manage resource allocation of a high level of abstraction and at the same time to establish guarantees for throughput and latency. Note that the goal remains constant for the duration of the query execution.

In addition, and regardless of the actual balance of the optimization goal, the cost-based optimizer always aims to prevent resource shortage (processing stalls, out-of-memory) and back-pressure in the processing pipelines, allowing for the unobstructed and continuous execution of the stream topology.

For accomplishing these tasks, the cost-based optimizer determines the resource requirements of every stream operator, based on the operator's algorithm and configuration, and on the data characteristics of the operator's input streams. Cost estimation assumes correlation of data characteristics, i.e. of the characteristics of the flow of stream events, where a finite set of events from the input streams generates a finite set of output events. Formally, the data characteristics of a data stream s are composed of an event rate r_s and a temporal window size w_s. So, the cost function of any operator $n \in N$ with i input streams $s_1, s_2, \ldots, s_i$ a parallelism of p, is calculated as

$$\mathrm{Cost}_n(\{r_{s_i}, w_{s_i}\}, p) = (\mathrm{cpu}_w, \mathrm{size}_w, r_{out}, w_{out})$$

where cpu_w denotes the computation complexity of n for handling one temporal window, and size_w corresponds to the storage complexity that n requires internally to manage the state of that window. Computation complexity, storage complexity, and output rate serve as indicators for resource requirements, as well as for latency and throughout estimation. We provided an estimation of each operator, working together with domain experts that detailed the values.

With this estimation, the optimizer can associate the operators with resources, by (1) aggregating operators into chains, which are deployed into individual slots for execution, and by (2) setting the chain

parallelism, such that replica of a particular chain are executed in separate slots.

With this approach, it is possible to propagate the costs bottom-up through the DAG that represents the query. The data characteristics of data sources (DAG leaves) are determined before cost estimation can start. Note that the values of each data characteristics of data sources may vary over time due to the amount of incoming data. Ideally, the data source characteristics are known upfront, e.g. consulting statistics from previous accesses. Alternatively, sampling is applied to gauge the characteristics of the current data flow. Finally, the API allows the DSP programmer to manually set/override the data characteristics of individual data sources.

The basic strategy of optimization is the following: Start at the DAG leaves and create chains initially containing only a data source. Eventually each chain C will be allocated to a slot, an execution container, which is a unit for resource management, having a single CPU core and a fixed amount of memory, hence any chain's resource requirements may not exceed the resource capacity offered by its allocated slot. Recursively try to extend the current chain across the subsequent operator n, by checking the following conditions.

1. n is chainable
2. n supports the parallelism of C
3. Chaining creates no back-pressure
4. Chaining creates no resource shortage (e.g. CPU / memory of execution container)
5. Chaining complies latency constraint of optimization goal
6. Chaining complies throughput constraint of optimization goal

As a consequence, new chains are started whenever a conflicting condition is detected. If all conditions apply, the chain is extended, thereby minimizing the total number of chains in a topology, and ultimately controlling the cost of execution.

The amortized complexity of cost-based stream optimization is $O(|N|\log|N|)$, as it corresponds to one traversal of the DAG with eventual backtracking for adapting chain parallelism.

As mentioned before, the characteristics of data sources are not constant while stream topologies tend to be deployed for a long period of time. In the following we describe how monitoring continuously revalidates the topology against the optimization goals and re-deploys the topology if significant changes are detected.

4.4.2 Monitoring System

Apache Flink exposes by default a number of metrics on the cluster usage, both for the master node (Job Manager) and for the slaves (Task Manager). It is possible to receive these metrics through JMX, and we store all the available metrics in Apache Lucene [Apa19c], which allows the user to perform range queries, which we use for getting the history of the needed metrics. We take into consideration the metrics related to the amount of resources used (e.g., RAM, CPU), and we monitor possible problems (e.g., back-pressure). But among all, the most important for our algorithm is the knowledge of the input and output rate, in both forms of size and number of events. The collected metrics are then forwarded to the feature extractor, which processes them to provide the cost-estimation the right parameter to actually check the redeployment of the topology.

4.4.3 Incremental learning for Dynamic Cost Estimation

Our solution for dynamic cost estimation unfolds as a learning problem. We formulate it as such to offer a generic, robust and flexible approach suitable for such massive distributed systems.

Dynamic cost estimation is an incremental process. Starting from an initial topology our system employs a machine learning algorithm that uses recent history to build a model of the data (i.e. query specific data: input rate, window size; goal: cost, latency, throughput; topology) and its evolution to estimate predictions. Such a model is updated for each new query, such that the model evolves with the data it represents and is able to accurately trigger a topology reconfiguration. Moreover, this kind of model needs to capture and accommodate the changes in the process generating the data (e.g. learning the correlation among input rate and the topology) to either generate

a prediction conditioned on history or to trigger a full model retraining. Of course, this implies either the use of incremental algorithms or the periodic retraining with batch algorithms (expensive in terms of time and resource consumption - critical in such a dynamic cost estimation problem). As a first implemented method, we take care of the history of the input rate for the operator, we perform the average, and we send it to the cost-estimation function for building the new topology.

In addition, we propose a new approach using incremental learning, which is a learning paradigm where computations adjust to any external change to their data automatically. As is it the case in online machine learning, applications need to respond to incremental modifications to data (i.e. update the topology based on the desired cost, throughput and latency). Being incremental, such modifications often require incremental modifications to the output, making it possible to respond to them asymptotically faster than recomputing from scratch.

In such cases, taking advantage of incremental behavior, dramatically improves performance, especially as the system evolves in time for subsequent queries.

In order to decide on a topology change, the Feature Extractor component receives as input the query and the stream parameters (i.e. input rate and window size), the goal (i.e. the desired cost, latency, and throughput) and the measured metrics (i.e. usage, measured input rate etc.) and when needed triggers a topology change through to the cost-estimation function. In order to make the solution flexible and adaptive, we extend from a static policy switch to an incremental learning approach. Such an approach assumes learning the dependencies among the input and output variables of the cost-estimation function, in a pairwise fashion to exploit all the underlying correlations among the measured metrics of the current topology and the current query parameters, for example.

Learning such pairwise functional dependencies among the variables is basically a regression problem. Various methods for both univariate and multivariate regression have been developed, yet it is not trivial how to extend them to cope with the evolving nature of the problem. In other words, there is not a straightforward approach to incremental regression.

In order to explore the underlying relations among the variables in our problem, we started by exploring a linear regression problem, employing a simple incremental Linear Least Squares (LLS) model (described in Figure 4.4). The linear least squares fitting technique is the simplest and most commonly applied for linear regression and provides a solution to the problem of finding the best fitting straight line through a set of points. It tries to minimize the sum squares of the deviations of a set of n data points:

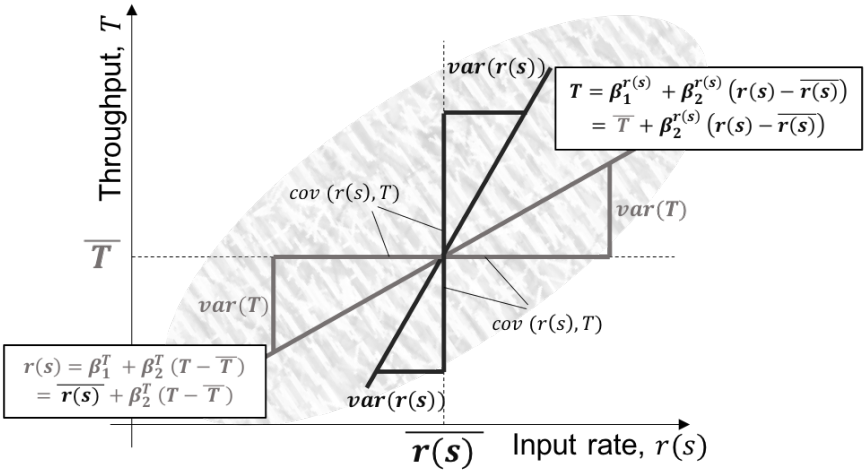

Figure 4.4: Linear Least Squares (LLS) model example

In our case, for example, for the input rate to throughput dependency we can incrementally calculate:

- mean: $\overline{r_n(s)} = \overline{r_{n-1}(s)} + \frac{1}{n}((s) - \overline{r_{n-1}(s)}$, where n is the window size, so $\overline{r_{n-1}(s)}$ is the mean over $n-1$ readings (without the current reading), and $\overline{r_n(s)}$ is the mean of window with the current read

- variance (2^{nd} moment):
  $m_{2,n} = m_{2,n-1} + (r_n(s) - \overline{r_{n-1}(s)})(r_n(s) - \overline{r_n(s)})$

- covariance:
  $s_{r(s)T,n} = \frac{n-2}{n-1} s_{r(s)T,n-q} + \frac{1}{n}(r_n(s) - \overline{r_{n-1}(s)})(T_n - \overline{T_{n-1}})$

In general, the nonlinear regression problem assumes a loss function $L^2 = \sum[T - f(r(s), \beta_1, \ldots, \beta_n)]^2$. In particular, for the linear case we have:

$$L^2 = \sum[T - f(r(s)\beta_2 + \beta_1)]^2$$

We learn incrementally the coefficients:

$$\beta_1 = \overline{T} - \beta_2 r(s) \text{ and } \beta_2 = \frac{s_{r(s)T,n}}{m_{2,n}^2}$$

Such a model uses incrementally calculated descriptive statistics and is able to cope with the dynamic updates of topology in our solution. At the moment, the model is considering linear functional relations among the variables, for example, window size and topology equivalence class. For more complex, nonlinear dependencies such a model will fail to capture the underlying relations. In order to cope with such a problem, the regression mechanism could be extended to incrementally approximate also the nonlinearity dependencies among the variables. For example, we are planning to use an incremental Support Vector Machine (SVM). Support vector machines (SVMs) are supervised learning methods used for classification, regression and outliers detection. Among the advantages of support vector machines are: the effective in high dimensional spaces, such as the dynamic cost estimation for topology reconfiguration; the effectiveness in cases where number of dimensions is greater than the number of samples; the use of a subset of training points in the decision function (i.e. support vectors), so it is also memory efficient and suitable for dynamic cost estimation, and can learn highly nonlinear dependencies by using nonlinear kernels to encode the input data.

Given training vectors $r_i(s) \in \mathbb{R}, i = 1, \ldots, n$, and a vector $T \in \mathbb{R}$, SVM solves the following optimal problem:

$$\min_{w,b,\zeta,\zeta^*} \frac{1}{2} w^T w + c \sum_{i=1}^{n} (\zeta_i + \zeta_i^*)$$

$$\begin{aligned} \text{subject to } T_i - w^T \phi(x_i) - b &\leqslant \epsilon + \zeta_i, \\ w^T \phi(r_i(s)) + b - T_i &\leqslant \epsilon + \zeta_i^*, \\ \zeta_i, \zeta_i^* &\geqslant 0, i = 1, \ldots, n \end{aligned}$$

where ζ_i, ζ_i^* are the slack variables and $\phi(r(s))$ is the kernel. Here training vectors are implicitly mapped into a higher dimensional space by the function ϕ.

Training a support vector machine (SVM) requires solving a quadratic programming (QP) problem in a number of coefficients equal to the number of training examples. For very large datasets, standard nu-

meric techniques for QP become infeasible, this is why an incremental approach is definitely an approach to follow. As an extension to our linear regressor using LLS, we will propose an on-line, incremental SVM alternative, that formulates the (exact) solution for n+1 training data in terms of that for n data and the current data point (i.e. measurement). The incremental procedure would also be reversible and allow "unlearning" of each training sample to remove the impact it had on the dynamic estimation of the functional relations in the topology modifications decision.

The ultimate goal of the incremental learning component of our system is to learn the relevant pair of variables which have a strong contribution to the decision to change the topology. This assumes learning pairwise functions among them while making sure that consensus is reached among any variable. Such an approach assumes building a network (i.e. a graph representation) in which each vertex is a variable (i.e. input rate, window size, topology parallelism, topology equivalence class entries etc.) and connections among vertices (i.e. edges) represent the functional relationship connecting those variables, as learned by LLS and SVM.

Such a method would use an entropy reduction technique [ARC16] to actually select which variable pairs are the most informative and have a strong correlation. The system would allow to actually have a consistent state in the estimation process, i.e. all relations would be satisfied and the topology adjusted accordingly. A sample depiction of the system is provided in Figure 4.5.

This system is composed of a pipeline which feeds time-series sensory input; compute statistics for individual and pairs of sensors (entropy and mutual information); computes statistical distance and conditional entropies to extract statistical relatedness; creates a connectivity array using entropy reduction (minimization), as shown in the left panel. In the right panel, the underlying functionality behind the correlation learning is depicted.

Such an approach will enable the dynamic cost estimator to learn which of the available variables are correlated, learn the pairwise relations among them and then use these learned relations to take a decision on the topology update policy based on the consensus of the learned functions.

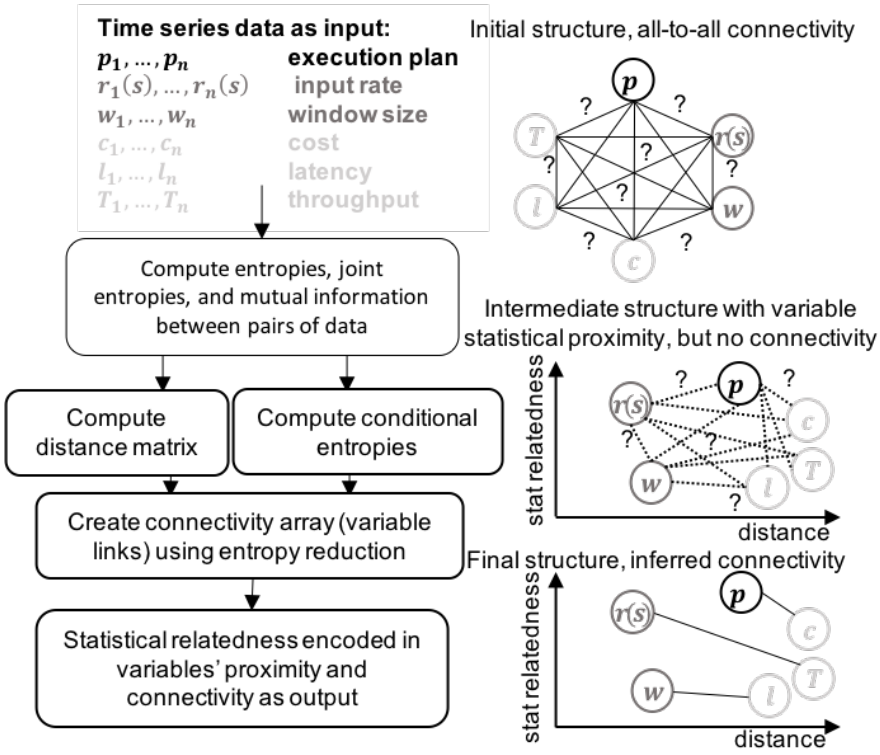

Figure 4.5: Entropy reduction technique for correlation learning

4.5 EXPERIMENTS

We present here a set of experiments to validate our framework for a dynamic cost estimation of the resources given the user goal. We show the results with respect to the static cost estimation and to Apache Flink by itself, describing the advantages and disadvantages of our system. We analyze both latency and throughput, which are the most used metrics to check the features of a system.

[System Description] The experiments were run over a cluster of 4 machines running RedHat 6.5. All the machines were featured with 126 GB of main memory and 24 cores. We perform our experiments with the standard version of Apache Flink 1.4.2, and we apply our framework on top of it. *base* is the original 1.4.2 version, while in *static* is enabled the static estimation at the deployment of the application, and *dynamic* describes the performance of the enabled dynamic goal-oriented cost-based estimation. If *static* performs a standard starting estimation, with *dynamic* is running another application which actually takes care of the rescheduling of the jobs. We perform our evaluation with the TPC-H benchmark [TPC19], creating the data

through the data generator and applying one of the queries they propose. We tested our framework with a constant input rate of 4000 elements/second for 2 hours.

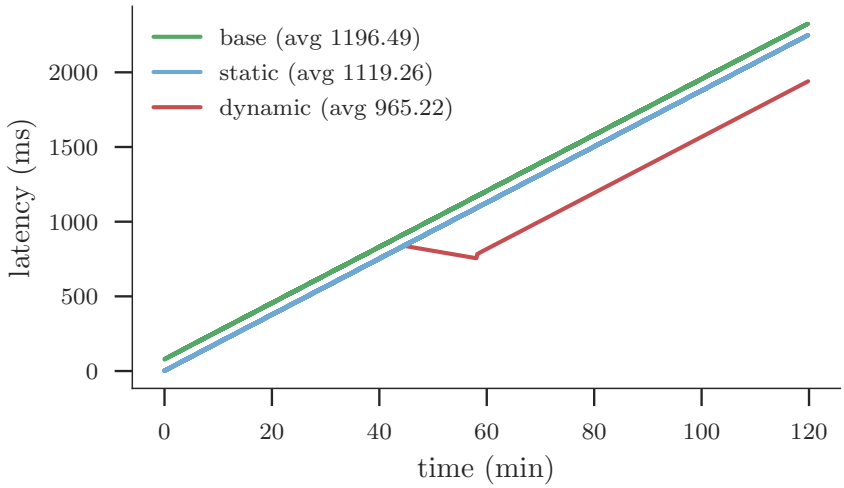

Figure 4.6: Latency evaluation for TPC-H query

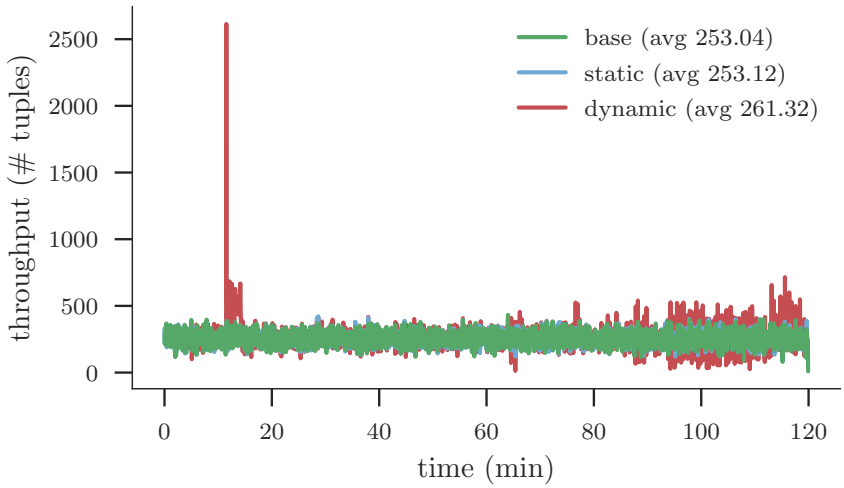

Figure 4.7: Throughput evaluation for TPC-H query

Figure 4.6 and Figure 4.7 show the performance of our framework with the TPC-H query number 3, which is listed below.

```
SELECT
    l_orderkey,
    sum(l_extendedprice * (1 - l_discount)) as revenue,
    o_orderdate,
    o_shippriority
FROM
    customer,
    orders,
    lineitem
WHERE
    c_mktsegment = 'BUILDING'
    AND c_custkey = o_custkey
    AND l_orderkey = o_orderkey
    AND o_orderdate < date '1995-03-15'
    AND l_shipdate > date '1995-03-15'
GROUP BY
    l_orderkey,
    o_orderdate,
    o_shippriority
ORDER BY
    revenue desc,
    o_orderdate
LIMIT 20;
```

The former presents an analysis on the latency, while the latter on the throughput of the query. For what concerns the *dynamic* estimation, we optimized the latency for the first plot, while for the throughput figure, the optimization goal was the throughput. For the latency chart, first of all, we see that the latency is high, and it is still increasing. That is because the number of elements sent to Flink was high for such a query, and its execution was computationally intensive. But this is not a problem caused by our implementation since even the base version reports the same behavior. We can see that the *static* estimation is giving a small advantage w.r.t. the *base* version, which proves that the cost function we implemented is a good starting point.

Moreover, the *dynamic* estimation shows better performance after 45 minutes, when the job was rescheduled. We were using a constant input rate, so the job was rescheduled only once. Moreover, a fluctuating input rate leads to even bigger, since the job resources follow the input rate.

For what concerns the throughput, we see that both the *static* estimation almost follows the *base* version, which is stable during all the execution, but with a slightly higher the number of output tuples per second. The *dynamic* estimation shows the same behavior of the *static* up to the rescheduling (after 10 minutes). Then, it presents a peak, due to the new configuration that allows it to output the queued elements. After that, it maintains a constantly higher throughput for the remaining execution time.

4.6 SUMMARY

In this chapter, by focusing on the goal metric the user needs to optimize, we propose a solution to give the results that the user wants. Hence, we provide qualitative data considering the user needs, which means that we contextualized the quality of the data by prioritizing the optimization goal defined by the user in a streaming environment. In particular, we considered the problem of dynamically allocating the resources for a streaming application in a cluster environment. The novelty of this approach is that the amount of allocated resources is bound to a goal defined by the user, which consists of a value for cost, latency, and throughput. Moira, the framework we propose, is a system built on top of Apache Flink, which, from an initial static estimation of the user query that deploys the first execution plan, dynamically accesses the metrics exposed by Flink and the characteristics of the incoming data. The system will then use both the metrics and the incoming data characteristics as features to reach the user goal. If the new cost-estimation suggests the deployment of a new plan, Moira actually reschedules the job with the new topology.

5 DBMATCHER - GIVING VALUE TO THE DATA

If in Chapter 3 the focus was on the task applied over the data, in Chapter 4 it was on the user needs that specifies a goal for an application, in this chapter we contextualize the data by itself. We present an application for entity resolution that profiles the data to understand the intrinsic value it has. While existing approaches focus mainly on ad-hoc solutions or needs the domain expertize to obtain high valuable results, the approach we propose aims at finding the similarity metric that leads to the best results for the available data.

5.1 CONTRIBUTIONS AND OUTLINE

In the following, we highlight the contributions of this chapter and present how it is organized. We provide a motivation for the need for such work (Section 5.2). (i) We extend the notion of entity matching by adding the definition of profile (Section 5.3). (ii) Section 5.4 highlights the components of the framework we developed to enhance the performance of entity-matching systems and allow the users not to be a domain expert for running such a system. (iii) The internal of the components of the framework, i .e., the partitioning of the data, the computation of the profile, and the similarity matcher, are presented in Section 5.5, together with the three algorithms we propose. (iv) We detail how the idea of profile could be integrated with running systems by applying it as a blocking technique, to speed up the performance (Section 5.6). Then, we show in the experimental section the preliminary results about the usage of this framework (Section 5.7).

ID	Name	Age	Birth Place	Job
1	Alan	24	New Yorke	—
2	William	36	Londron	—
3	Bob	87	New Orleans	Retired
4	Bob	18	Mexico City	Student
5	Bill	35	London	Researcher

Table 5.1: A portion of the dataset about people living in London.

5.2 MOTIVATING EXAMPLE

Consider a data analyst that has been asked from the municipality of London to perform some analytics over the people living in the city. Figure 5.1 presents a portion of such data. As can be seen, the tuples available for the researcher may not be clean, e.g., the tuple with `ID=1` has a typo in the `Birth Place` column. The dirtiness may be due to misspellings, missing information, or, even worse, wrong information.

The first task performed by the analyst is to look for duplicates in the dataset, which is the case of the tuples with `ID` 2 and 5 (i.e., Bill is the short version of William, 35 is a typo and, it should have been 36, and Londron contains a misspelling). Hence, there are two options to perform duplicate detection. The first is about asking some experts of the dataset which would be the best metric (or metrics) to identify the redundant tuples, while the second option does not need the involvement of an expert, but it executes of a bunch of algorithms and then it merges their results. However, the drawback of the first option is the need for a domain expert, which is not always available. On the other hand, the second option of running a bunch of metrics altogether is computationally expensive, since running multiple measures needs time.

Hence, there is the urgency for a framework that both learns when a specific metric gives promising results (what the expert does in such scenario) and has good performances (i.e., takes less time than running the bunch of metrics mentioned above). We identified that this could be possible by dividing the dataset based on the common characteristics of the records. These common characteristics are determined by a profile, which would determine the metric that gives the most impressive results on such a portion of the data. For example,

the framework identifies the first partition composed by the tuples with IDs 1, 3, and 4 because they have multiple words in the value of the Birth Place attribute. The second partition is given by those values that have a single word in the same attribute, which are the tuples with ID 2 and 5. The system has learned that for elements with multiple words, the best metric is the Jaccard similarity, while for values with a single word and numbers is the Edit distance. The system retrieves that the distance between tuple with ID 2 and 5 is not that hight. Thus, they are two match candidates. On the other hand, on the other partition, none of the tuples is suitable to be a match.

The need for a framework that learns how to optimize the results while applying the correct similarity measure to the correct portion of data is vital. Thus, such a framework would follow the hints provided by the dataset profiles and will run only the metric suggested, to minimize the running time of the system and maximize the accuracy of the results.

5.3 PROBLEM STATEMENT

Consider a dataset D, which is a set of structures modeling some real-world scenario, where the set of all possible datasets is denoted by $\mathcal{D}$. We define a tuple as an element $t_i \in D$, with $0 \leqslant i \leqslant |D|$ that represents a single entity belonging to the dataset D. Hence, a set of tuples composes a dataset. There exists the set of all possible attributes or properties $\mathcal{P}$ and each entity contains information about a set of attribute $P \subseteq \mathcal{P}$. Then, we assume the existence of a set of values $\mathcal{V}$, which also represents any possible value of any attribute $p \in \mathcal{P}$.

Thus, we state that a dataset D has a set of attributes $P \subseteq \mathcal{P}$. Hence, a tuple $t_i \in D$ representing the i-th tuple of D has a value t_i^p for each property $p \in P$.

Example 3. Figure 5.1 presents a sample of a dataset about people living in London. It reports a set of attributes P composed by ID, Name, Age, Birth Place, and Job. For instance, tuple t_3 is the one with ID = 3, which has $t_3^{\text{Name}} = \text{Bill}$ and $t_3^{\text{Age}} = 87$.

In such context, the goal of entity-matching is to find tuples that actually report information about the same real-world entity. This investigation is usually performed through similarity measures, which give a score about the similarity of two tuples.

Definition 9. A similarity measure is a function $sim : \mathcal{V} \times \mathcal{V} \rightarrow [0, 1]$, with $v_a, v_b \in \mathcal{V}$ that takes as input two values and outputs a score between 0 and 1. In the same way, we define the distance function as $dist(v_a, v_b) = 1 - sim(v_a, v_b)$, with again $v_a, v_b \in \mathcal{V}$.

The set of all the similarity measures is denoted by $\mathcal{S}$. When a similarity measure is applied to two data values and their output is close to 1 (or a distance close to 0), it means that the two values are almost identical, while their closeness decreases (distance increases) when their similarity score is close to 0 (distance score is close to 1). Different similarity measures resolve in mismatching output values for the same two values, which intuitively means that two similarity measures behave differently.

Definition 10. Entity matching is a function $EM : \mathcal{S} \times \mathcal{D} \times [0, 1] \rightarrow \mathcal{D} \times \mathcal{D}$ that given a similarity measure $sim \in \mathcal{S}$, a dataset $D \in \mathcal{D}$, and a threshold t between 0 and 1 returns a set of pairs that match.

The matching elements are those pairs of values in D that, if compared with a similarity sim, output a value that is higher than the defined threshold t. Starting from the observation raised above, different similarity measures give different results, thus, when applying various similarity metrics over a complete dataset, they will lead to various output.

Hence, this brings one question: (Q1) which is the best metric to apply on the data? Usually, a domain expert should reply to this question presenting the correct way to handle the data. However, it is not always available. To provide a solution, we identified the profile of a dataset as a mean for understanding the goodness of a metric for the data it contains. A profile is a feature vector that describes the portion of the data it contains. Each profile produces a score k when assigned to a similarity that measures how well that profile performs for that similarity.

An improvement is to apply to multiple portions of the dataset different metrics, in order to improve the quality of the outcomes,

Algorithm 4 DBMATCHER - OFFLINE PHASE

Require: A dataset D, a set of similarities S,
a ground-truth of duplicates gr
Ensure: The training of the system model

```
1: results ← loadOrDefaultResults(default=map())
2: profile ← computeProfile(D)
3: for all similarity ∈ S do
4:     duplicates ← applySimilarity(D, similarity)
5:     performance ← checkResults(D, duplicates, gr)
6:     results.put(key=⟨profile, similarity⟩,
                   value=performance)
```

but not to apply several metrics on the same data. Thus, to find a matching we need a function c from the set of all the functions with the same signature $\mathcal{C}$, such that:

Definition 11. $c(D, n) \rightarrow (2^D \times sim)^n$ is a function that given a dataset $D \in \mathcal{D}$ and a positive natural number n, creates a set of n portions of the dataset (i.e., subsets), with a similarity function assigned to each portion.

There are multiple functions with this signature, so we define the set of all this functions as $\mathcal{C}$ and we introduce a cost function $f : (2^D \times sim)^n \rightarrow w$. This function f takes as input a set of portions, each assigned to a similarity measure, that is the output of the function c, and returns a score w. This score measures the sum of the distance of the profile of each portion from the most similar that is known to produce a high score k for the related similarity. This is needed to partition the dataset in the best possible way, which means in such a way that the distances of the profiles of the created partitions from the profiles of the partitions already known is minimized.

We aim at minimizing the value of the function f in order to find the best dividing function c for applying a specific function to each subset of the dataset. Thus, we formalize this minimization problem as such:

Definition 12. $\underset{c \in \mathcal{C}}{\arg\min}\, f(x) := \{c \mid \forall x \in \mathcal{C} : f(c) \leqslant f(x)\}$

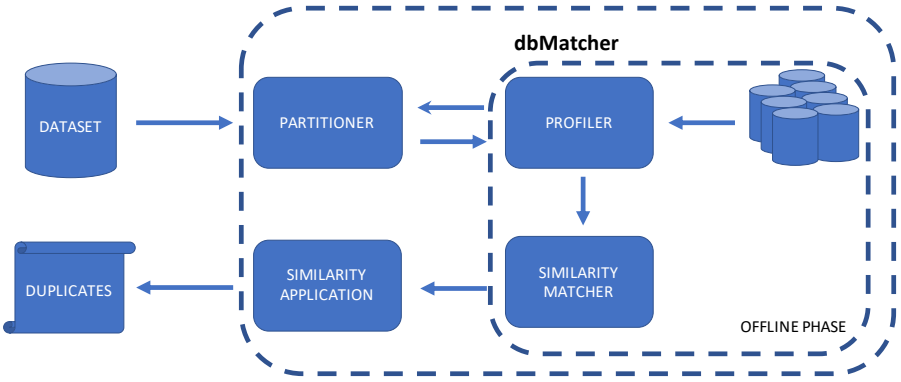

Figure 5.1: dbMatcher overview

5.4 DBMATCHER OVERVIEW

In this section, we discuss the modules that compose the framework for running the most promising similarity metric through the creation of a profile of the dataset. The framework, called dbMatcher, implements a scenario that satisfies the data model previously defined and answers the open questions raised in the same section (Section 5.3). The system is depicted in Figure 5.1, which shows the main modules the framework comprises: the partitioner, the profiler, the similarity matcher, and the similarity applier that actually performs the comparison.

Such framework works in two phases, an online process that efficiently and effectively finds the duplicates in a given dataset, and an offline phase that is run first and is needed by the online phase to be able to obtain such results.

Algorithm 4 represents the pseudo-code of the offline phase. We will briefly describe the functioning of the framework in this section, while in the following, we will deeply analyze the functions that are underlined. The system receives a dataset and comes with a predefined set of metrics, which can be expanded, adding extra measures that are of interest for the user. If the training phase has already been executed, the previously computed results are loaded. Otherwise, they are initialized as an empty map (line 1). Then, the system creates the profile for the given dataset (line 2), and then the collected results are tested with all the available similarities (line 3-4). The obtained duplicates are checked against the ground truth duplicates provided as input (line 5). The information about the performance of the dif-

ferent datasets are collected in the results structure and will serve as the training set for the similarity matcher (line 6). Ideally, the offline phase should receive as input and process a set of new datasets at the beginning, then the online phase works with the learned parameters by the offline phase.

On the other hand, the representation of the online phase can be found in Algorithm 5. During the online phase, the system takes as input a dataset provided by the user and processes it. The first module invoked is the partitioner, which knows the profiles already sought by the matching profiler in the offline phase, without being aware of the metric to which it will be matched. The partitioner splits the dataset in such a way that each partition determines a profile that is as close as possible to a profile known (line 2). Such an approach will maximize the overall performance of the system, both in terms of running time and accuracy of the results. We will discuss in details how the partitioner works in Section 5.5.1, presenting three approaches we implemented, the greedy, the incremental, and the adaptive approach.

Next, the profiler actually takes the partitions generated by the partitioner and creates a profile for each partition (line 3-4). A profile is basically a description of the dataset, that summarizes the records it contains. The profile can contain, for example, information on the type of the data, e.g., strings, numbers, dates, or objects, and the results of some aggregations performed on the records, e.g., maximum value, average value, or string length. It is represented as a feature vector that stores, in each position of the vector, specific information. Our implementation of the profile is provided in Section 5.5.2.

The similarity matcher is the component of the framework that determines the application of the similarity measures to the partitions it takes as input. Given the profile of each partition just computed by the profiler, the similarity matcher aims at finding the similarity measure that works better (in terms of accuracy of the results) for a partition with such a profile (line 5-6). Section 5.5.3 presents the algorithm used for this matching part.

The last block in dbMatcher is the similarity applier. It receives a partition, which contains a set of records, and a similarity measure to apply among them (line 6 and lines 7-9). The output of this module is the set of the entities of the original dataset that are recognized to

Algorithm 5 DBMATCHER - ONLINE PHASE

Require: A dataset D, a number num of partitions, and a threshold ϵ
Ensure: A set of tuple $duplicates$ found in D

```
1: duplicates ← set()
2: partitions ← partitionDataset(D, num, ε)
3: for all partition ∈ partitions do
4:     profile ← computeProfile(partition)
5:     simFunction ← matchSimilarity(profile)
6:     duplicates.addAll(
           applySimilarity(partition, simFunction))
7: return duplicates

8: function applySimilarity(partition, simFunction)
9:     return simFunction(partition)
```

be duplicated. The whole process enables an easy parallelization of the similarity metrics since the set of the partitions determined by the partitioner do not consider overlapping portions.

5.5 INTERNALS

In this section, we provide the details of each module of dbMatcher, the framework generally described in the previous section.

5.5.1 Partitioner

The first module involved in the process is the first we dive into. The partitioner takes a dataset, and splits it in multiple partitions. The key idea behind this component is that it needs to maximize the accuracy of the results that would be obtained based on the knowledge discovered by the similarity matcher. It means that the partitions generated by this component need to produce profiles as much as similar to those that has been learned to lead to good results. Hence, we propose three algorithms to actually split the dataset into multiple partitions.

Algorithm 6 presents the pseudocode for the first and most inaccurate method we propose, the greedy approach. It takes a dataset as input and splits it using as a baseline a k-mean clustering algorithm,

Algorithm 6 PARTITIONDATASET - GREEDY APPROACH

Require: A dataset D, a number num of partitions, and a threshold ϵ
Ensure: A set of num partitions P

```
1: P ← set()
2: distance ← ∞
3: distMap ← dict()                ▷ key: the partition, value: its distance
4: while distance > ε do
5:     if Not distMap.isEmpty then
6:         partition ← distMap.getKeyWithMinValue()
7:         if distMap[partition] < ε then
8:             P.add(partition)
9:             num ← num − 1
10:    missing ← NOTALREADYASSIGNED(D, P)
11:    if missing.isEmpty then
12:        return P
13:    tmp ← CREATERANDOMPARTITIONING(missing, num)
14:    distMap ← MEASUREPARTITIONINGDISTANCE(tmp)
15:    distance ← sum (distMap.values)
16: raise NoSolutionError

17: function NOTALREADYASSIGNED(D, P)
18:    toReturn ← {}
19:    for record ∈ D do
20:        found ← False
21:        for partition ∈ P do
22:            if record ∈ partition then
23:                found ← True
24:        if found = False then
25:            toReturn.add(found)
26:    return toReturn
```

with k as the number of partitions we aim at finding. In general, it has to be noticed that the higher is the number of the partitions and the lower is the threshold, i.e., the sum of the differences of the profiles already computed to the profiles already known from the offline phase, the higher is the chance to get better performance in terms of computing time and accuracy of the results. The greedy method measures the distance from the profile of each partition to the profile already computed in the offline phase that is the most similar. If the sum of such distances for the randomly selected partitions is above the threshold defined by the user, then the greedy method keeps the best partition (i.e., the one that has the lowest distance), and discards the others. This happens only if the distance of the profile of the kept

Algorithm 7 PARTITIONDATASET - ITERATIVE APPROACH

Require: A dataset D, a number num of partitions, and a threshold ϵ
Ensure: A set of num partitions P

```
1:  P ← createEmtpy(num)
2:  for record ∈ D do
3:      score ← ∞
4:      mostConvenient ← null
5:      for partition ∈ P do
6:          p ← partition ∪ record
7:          distance ← measurePartitioningDistance(p)
8:          if score > distance then
9:              score ← distance
10:             mostConvenient ← partition.id
11:     P[mostConvenient] ← P[mostConvenient] + record
12: return P

13: function createEmtpy(n)
14:     toReturn ← {}
15:     while n > 0 do
16:         toReturn.put(key =n, value = set())
17:         n ← n - 1
18:     return toReturn
```

partition is below the threshold, otherwise, if the distance is higher than the threshold, we discard everything, and we start from the beginning. Then, the greedy approach randomly selects the subset of the original dataset composed by the records in the discarded partitions and randomly splits it in n partitions, where n is the number of partitions defined by the user minus 1 (the partition we kept). This greedy algorithm dynamically decreases the distance of the profiles, in such a way that the accuracy of the results increases.

The second method we propose is an iterative approach, detailed in Algorithm 7. We start from a set of num empty sets (line 1). Then, we take a record at time from the given dataset and we check for each partition what would be the benefits of adding the record to that partition (line 5-10). The partition that with such new element would have the profile that has the least distance from a profile already computed would be the candidate, and we will add the record to that partition (line 11). Then, we move to the next tuple, we apply the same process, until no records are left, such that each element has been assigned to a partition. This would enable the maximization of the similarity to the already computed profiles in the offline phase.

Algorithm 8 PARTITIONDATASET - ADAPTIVE APPROACH

Require: A dataset D, a number num of partitions, and a threshold ϵ
Ensure: A set of num partitions P

```
1:  P ← CREATERANDOMPARTITIONING(missing, num)
2:  for partition in P do
3:      outliers ← GETOUTLIER(partition)
4:      partition.removeAll(outliers)
5:      for record ∈ outliers do
6:          score ← ∞
7:          mostConvenient ← null
8:          for partition ∈ P do
9:              p ← partition ∪ record
10:             distance ← MEASUREPARTITIONINGDISTANCE(p)
11:             if score > distance then
12:                 score ← distance
13:                 mostConvenient ← partition.id
14:         P[mostConvenient] ← P[mostConvenient] + record
15: return P
```

The third and last approach we discuss is the adaptive one, described in Algorithm 8. It can be considered as the aggregation of the greedy and the iterative approaches, taking the advantages of both. We do not start from scratch as it happens in the iterative approach, but we get a default random partitioning that we will refine as the iterative approach does. Hence, from the default random partitioning (line 1), we iterate over these partitions and we find the outliers in each of them (line 2-3). We define an outlier as a record that worsens sensibly the similarity of a profile from an already computed profile. Then, we remove the found outliers from the corresponding partition (line 4) and we start the iterative process. For each of these outliers that have been removed from the initial partition, we iterate over the other partitions and we find the one that would have the higher benefits by adding the analyzed record (line 5-13). Once we found it, we assign the record to such partition (line 14), and we move to the next record defined as outlier for the current partition. Then, we move to the next partition, until all the partitions are processed and we found the best partitioning.

5.5.2 Profiler

The module that actually contains the core logic of the process is the profiler. As it was introduced in the previous section, the profile is composed by a set of features, which describe the dataset analyzed (full dataset or portion it does not matter). Since a profile should summarize the data, we dived in the literature to understand how to describe a dataset. There are two main categories of approaches, a statistical-oriented method and an information-oriented. The statistical-oriented approach aims at filling the feature vector of analysis and statistics performed over the data. Hence, the feature vector would contain the aggregation results of some specific query [Abe+18]. The information-oriented approach applies instead a summarization of the dataset, e.g., for a dataset containing cities, a summary may state that "75% of the reported cities are in the U.S. and are populated by more than 1 million people" [JGP15].

In our implementation, we followed the statistical approach, because less affected by the errors that are present in the dataset, such as typos or missing information. In the feature vector we applied, first we collect information about the type of the data (i.e., integer, float, double, string, date, time), then, in case of strings, we perform a topic detection analysis to elements that report information of the same area (discovered through Latent Dirichlet Allocation - LDA). Moreover, we add to the profile of a dataset in case of numbers the average, the maximum, and the minimum value, as well as the standard deviation, while for string we consider the same aggregations referred to the length of the item. It is taken into account also the difference between the maximum and minimum value, for the actual number or of the length of the string, along with the number of missing information or nulls. We focused on these since we stated that such information would be a starting point for gathering the intrinsic value of the data.

5.5.3 Similarity Matcher

The last component we analyze in this section is the similarity matcher. It receives a profile and a threshold, and it checks whether

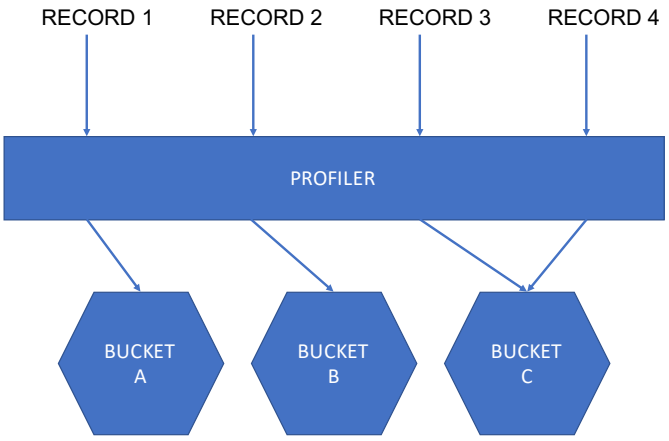

Figure 5.2: Profiler application as a pre-processing for blocking, to gather to which block a record should be added.

such profile matches an already analyzed profile diverging by at most the value of the threshold. Here, for such difference we just apply the euclidean distance on the two feature vectors of the two profiles. Such metric allows us not to focus on the measurement of the distance of two vectors, which is out of the scope of this work, but concentrate on the calculation of the distance, emphasizing on the application of the similarities. We enabled the usage of several similarities, taken from the Python library `py_entitymatching`, developed within the context of the Magellan project [Kon+16]. The similarity we used are: affine measure, Hamming distance and similarity, Levenshtein distance and similarity, Jaro and Jaro Winkler measures, Needleman-Wunsch measure, Smith-Waterman measure, Jaccard measure, cosine similarity, overlap coefficient, Dice score, Monge-Elkan measure, and exact match check.

5.6 BLOCKING APPLICATION

In this section, we will describe our intuition about the relationship between the idea of profile over the tuples and the blocking technique. As mentioned in the literature, blocking is a technique for reducing the number of comparisons between records in the dataset that do not match. Similarly, we can consider our approach as a blocking technique, where each partition is a block, and we perform comparisons only among tuples within the same block (partition). However,

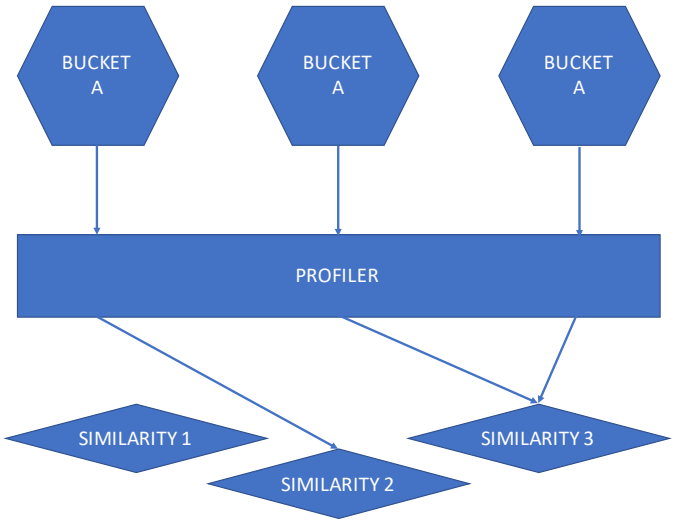

Figure 5.3: Profiler application as a post-processing for blocking, to gather which similarity measure would be better to use.

such analysis would be a limitation, since we also propose a robust framework to enhance the quality of the results while proposing the user the similarity metric (or the combination of metrics) to use in each block.

The application of the idea of the profile as a blocking technique can be performed in two different ways. The first is basically a pre-processing step for blocking, or better, the profile is the function for performing blocking. As depicted in Figure 5.2, this approach implies the computation of the profile for each record and hence assigning the tuple to the block corresponding to the most similar profile. In this way, we generate blocks with elements that share the same vector of features (i.e., the same profile). This approach can be considered similar to the application of a clustering algorithm over the dataset, where each point is characterized by the profile it has. However, such an approach may be highly costly since the profile can take a lot of time and resources to be computed multiple times.

The other option we specify is the chance to apply this idea, and hence compute the profile right after the blocking phase. Figure 5.3 graphically presents this post-processing, which enables the application of the correct similarity metric to the available data. With this approach, instead of having a single similarity for all the blocks, we assign a similarity to be used only and specifically on the set of records present in a specific block.

Name	Rows	Attribute
PRODUCT	4587	5
BEER	7343	4
RESTAURANT	862	6
MUSIC	62831	8

Table 5.2: Datasets Characteristics.

The application of both elements combined, the pre-processing and post-processing, leads to the privileges enabled by dbMatcher, the framework we presented in the previous section. Hence, this approach would allow the data practitioners to reduce the number of comparisons to perform by selecting the elements that together build a subset that matches a similarity known to perform well with such data. Then, it enables the application of such similarity to improve the quality of the results.

5.7 PRELIMINARY RESULTS

In this section, we present the preliminary results we obtained during the development of the dbMatcher framework. Since it is the starting point of the system, the main focus of this section is the offline phase, which aims at finding the similarity metric that gives the best results for a dataset. Hence, for learning this information we employed 4 datasets. Table 5.2 reports their information.

The PRODUCT dataset contains product obtained from Amazon and Google, the BEER contains data scraped from the websites BeerAdvocate and RateBeer, while the RESTAURANT reports data taken from Fodors and from Zagats, and the MUSIC records musical data from iTunes and Amazon. These datasets where used in the experimental evaluation of DeepMatcher [Mud+18]. This enables later a comparison of the results obtained with dbMatcher with those produced by DeepMatcher for evaluating the accuracy of the matches.

For the first step towards the evaluation of our framework, we took a set of similarity measures used in the literature [Kon+16], already listed in Section 5.5.3, and we applied them to the different datasets presented in Table 5.2. For each attribute of each of the datasets, we compared the pairs of elements with these similarities and we

Threshold	Precision	Recall	F1
0.1	0.101	1.000	0.184
0.2	0.101	1.000	0.184
0.3	0.101	1.000	0.184
0.4	0.101	0.999	0.184
0.5	0.100	0.978	0.182
0.6	0.100	0.920	0.181
0.7	0.144	0.616	0.234
0.8	0.331	0.281	0.304
0.9	0.514	0.125	0.201

Table 5.3: Results for Jaro similarity on `PRODUCTS` dataset over `title` attribute.

counted those retrieved as matching by the similarity, for several thresholds. We computed the confusion matrix of the obtained results comparing them against the available groundtruth.

As an example, we report in Table 5.3 the results of the Jaro similarity applied over the `title` attribute of the `PRODUCT` dataset, tested with multiple thresholds. Even if the length of the values in `title` is diverse since there are short and long names, the results show that the similarity metric produces the same results for the lowest three thresholds (i.e., 0.1, 0.2, 0.3), while the values start changing slowly from 0.4. The recall is obviously higher with lower thresholds since it produces less false negative defining as a match every pair that exceeds the threshold. On the other hand, the precision follows an inverse behavior with the false positives that decrease limiting the matching tuples. The table reports a low value for the F1 score, for all the threshold. It means that the Jaro similarity metric for the `title` column erroneously recognizes several matching pairs as non matching and viceversa. This tuning part would be essential for then proposing the correct similarity metric.

For these analyses, we focused on a single column, while the next step would be to understand how to integrate multiple columns, and subsequently how to manage their thresholds, eventually using different metrics for each column. A similar work has been performed for defining the right configuration for Raha, an error detection system [Mah+19].

Once we would be able to learn from the dataset the similarity metric that gives the best results, we would run it extensively with extra datasets, to then define the relation between the profile of the dataset

and the similarity metric. These steps would create a solid base for the offline phase. Afterward, for what concerns the online phase, we need to test the partitioning algorithms, described in Section 5.5.1, and perform a result comparison with other entity resolution frameworks.

5.8 SUMMARY

In this chapter, we focused on the data itself, measuring the intrinsic value it has. In particular, we presented a new technique to avoid the application of multiple similarity metrics altogether and the need of involving in the process a domain expert to have a consult about the similarity measure to apply. DbMatcher, the name of the framework we propose, implies an offline phase that allows the model to be trained. The system receives a known dataset, it computes the profile for such records, and then all the considered similarity measures would be applied on it and then their performance would be tested. Once the user has a new dataset, the system splits the dataset and computes the profile for each portion, in such a way that the similarity metric to apply to any portion would be done starting from trained model, for improving the results of the performance. We proposed three different metrics to partition the dataset and we present our approach to compute the profile. Moreover, this framework is an open system, where the user can plug her own partitioning or profiling technique. Then, we showed a possible application of the profile idea as a pre or post-processing technique for blocking, along with the benefits that such framework brings to the results with an experimental evaluation.

6 CONCLUSIONS

The quality of the data can be lead to serious consequences on the results of data analyses, such as misleading interpretations or errors. Moreover, we are in the Big Data era, and recent studies have shown that mainly poor quality data can be found in data lakes and large databases. Hence, given this need of qualitative data, the data quality topic has been studied deeply in the literature, in many different subfields. Generally, existing approaches mainly focuses on the evaluation of a specific characteristic of a dataset or they aim at cleaning it. Another field of the research presents methodologies that can be applied primarily in ad-hoc solutions or with the help of domain experts. However, existing approaches have not considered the actual usage of the data for computing the quality of the dataset itself. Moreover, providing an evaluation or an estimation of the quality of a dataset without knowing the need of the user that is using such data is a complex task and may not be worthy enough for the final user.

For these reasons, in this dissertation we studied new advancements on the topic of data quality, exploiting the concept of contextualizing the quality of a dataset. For this contextualization, we focused on three main aspects: (i) we inserted the task into the loop of the qualitative evaluation of the data, (ii) we highlighted the user need for the task execution, and (iii) we used the data itself and leveraged its intrinsic value to improve the performance of an entity resolution task.

In the following, we present the key contributions presented in this dissertation (Section 6.1) and the open challenges to which this work can lead (Section 6.2).

6.1 KEY CONTRIBUTIONS

In this dissertation, we studied the quality of the data by providing a new point-of-view: its contextualization. For us, it means to take into consideration other aspects other than the data. We focused on the task, on the user need, and on the intrinsic value the data have, and how each aspect can improve the performance of an application and the quality of its results. Hence, in the following, we describe the key contributions that each chapter of this dissertation brought.

6.1.1 Contextualizing the Task

We started our investigation from the contextualization of the task. We extended the notion of data quality by adding the task for which the data is about to be used. We consider the quality of a dataset not as the degree of the noise in it, but as a task-dependent function of the noise, presenting a framework that enables such computation, which is, to the best of our knowledge, the first of this kind. The framework we proposed is not bound to any specific data characteristic to evaluate, nor to any task or query. Usually, the proposed data quality metrics are static for a dataset, while we propose a dynamic metric that can also provide an estimation for similar datasets. We enabled this evaluation through the systematic introduction of different kinds of noise, taken from real world errors that can be found in the data. Then, we implemented different metrics, or we apply those that are already known for the given task, to measure the distance in the results that an error produces, and we finally provide a degree of the impact of the given error. We evaluated the approach over real and synthetic datasets, proving the advantages of the framework.

6.1.2 Contextualizing the User Needs

We continued the contextualization of the data quality by focusing on the user needs, in a streaming scenario. In a distributed system such as Apache Flink that can take a user job and deploy on a cluster, we presented a framework that, given a goal defined by the user, dynamically scales and allocates the resources to fulfill the goal. The

goal defined by the user can lead to a high throughput or a low latency of the results, or the minimization of the cost of the allocated resouces. To the best of our knowledge, this is the first work that does that. In that chapter, we provided a formal definition of the optimal solution for the user goal, being aware of usage metrics of the cluster, the incoming data, and the given user job. We show that the results of the framework outperforms the standard Apache Flink implementation both with a static and a dynamic plan.

6.1.3 Contextualizing the Data

Finally, our last contribution contextualizes the quality of the data by giving value to the data itself. We tackled the problem of entity resolution with a new approach that takes into account the profile the data has. We defined a profile as a description of the data, and we used that definition to improve the quality of the matches find by the framework we implemented. While other work consider a single similarity metric and usually that is suggested by a domain expert, our approach find the best one by computing the profile. We proposed three algorithms to partition the data and find the best similarity for each partition in order to maximize the output quality.

6.2 EXTENSIONS AND OPEN PROBLEMS

The studies conducted for this work along with the results obtained prompted us with some additional avenues for future studies. Hence, we describe the possible improvements and the challenges related to each study.

6.2.1 Contextualizing the Task

Quantifying Data Quality. We demonstrate that our work on the task contextualization, the fitness for use (Chapter 3), is able to measure how the changes in the data affect the task results. Suppose that a user is using our framework to select among two completely different datasets the one that fits better her needs. This would not be the

best choice, since we can have an indication, but we still have doubts about the decision to take, and that would be also time consuming since there is the need of running multiple times the framework on two different dataset. For these reasons, we would like to push a bit further and come up with a score that allows the user to compare two datasets, to identify the dataset that would fit better the task the user has in mind.

Data Cleaning. We mentioned that our framework gives a rank of errors that need to be cleaned, based on the impact they have on the performance of the task. However, the cleaning algorithm proposed are many. Hence, to help the user in this decision process, we would like to integrate already existent data cleaning algorithms to extend our application for enabling an automatic cleaning of the data. Moreover, we are considering to focus mainly on the identification of the portions of the dataset, i.e., subsets, that highly affect the results of the intended task. We can do it by integrating a sampling method for understanding the portion of the data that causes the higher degradation of the results. In this way, it would be easier and less expensive to improve the quality of the desired outcomes.

6.2.2 Contextualizing the User Needs

Smart Cities. In Chapter 4, we discussed the fulfillment of the user needs, by dynamically allocating the resources for a streaming job. We can think of it as a recommender system somehow. Changing scenario and moving to smart cities, the user may need, for example, to have custom suggestions about the route to take to arrive home, and what to visit in a new city. Hence, the management of traffic-lights enables such analysis since it has already to shorten the time spent in the traffic for a car, and it can suggest routes to match the user goal, which can be, for example, to visit new streets or to take the shortest path to save money. In addition, in the smart city it can be integrated a system for helping the users in their walking trips around the city, which can be for business, for visiting, or for shopping, among the others.

Anomaly Detection. The user may have the need to analyze the data it is processing. In particular, if we consider a streaming scenario,

one of the differences from a batch dataset is the ability to change the state and value of an element. The detection of such changes and their prediction is interesting from the research point of view, even if there are already works on it. In particular, having dynamic data may also change the definition of quality and change the user needs. For example, some works are discarding elements into a clustering algorithm based on their value. Then, the relevant values, those that have to be kept, has to be dynamically considered according to these changes.

6.2.3 Contextualizing the Data

Experimental Evaluating the Framework. Even if the framework is consistent and the algorithms have been developed, it is missing an experimental evaluation that proves the actual benefits of the system. Hence, the first step is to test with the datasets already mentioned in the preliminary results how the framework compares with the related work.

Profiling Data Quality. So far, we have computed the profile of a dataset only to gain information about the similarity metric to apply. If we apply our idea to another task, we would provide a profile that is not customized to the user needs since the user may be interested in some data quality characteristics and provide some functions of interest. Hence, there is the need for computing them efficiently and effectively, which our approach is not yet able to perform. Another challenge is the application of this process in a streaming scenario, with data that changes in real time. Monitoring the elements of interest for the user should not have any impact on the performance of the actual streaming framework. Moreover, the analysis should take into consideration the evolution of the data.

Data Lake Applications. Data Lakes are recently gaining a lot of attention. A new application could be to find related datasets among those available in the data lake. We may use our system to do it, but it would be easier to move the concept of profile to a new system. In general, similar datasets can be useful in multiple ways. For example, we may need more data to train our machine learning algorithm or as a source of extra information. This process would be a starting point

for then enabling the application of succeeding operations, such as entity linking, entity matching, or data integration among the others. Moreover, if we consider entity matching, identifying similar dataset (or portions of the dataset) allows a framework to understand which type of algorithm to apply to data of the same type. This would enable a parallelization and improvement of both the efficiency and the effectiveness of entity matching systems.

A DATA CHARACTERISTIC METRICS

[Nulls] A common technique is to measure the change in the number of nulls that are present in the results of the analytic task. To do so, we count the number of nulls for each attribute in the results of the original and the noisy datasets. Then, any distance metric can be used, e.g., the Jaccard similarity, to quantify the variation.

[Entropy] An alternative to the measurement of the nulls is the entropy [SW98] of each attribute, which is an indication of the information that is present in the results for a specific attribute. Given an attribute A and the set of the distinct values it contains $\mathcal{A}$, the entropy for A is:

$$\mathbb{H}(A) = -\sum_{a_i \in \mathcal{A}} p(a_i) \cdot \log(p(a_i))$$

where $p(a_i)$ is the probability of a value a_i, which is the frequency of the value a_i over the cardinality of A. With the entropy for every attribute for both the datasets, the distance is computed similarly to nulls.

[Value Cardinality] A third approach considers the cardinality of each attribute, i.e., the number of distinct values an attribute contains.

B BASELINE VALIDATION

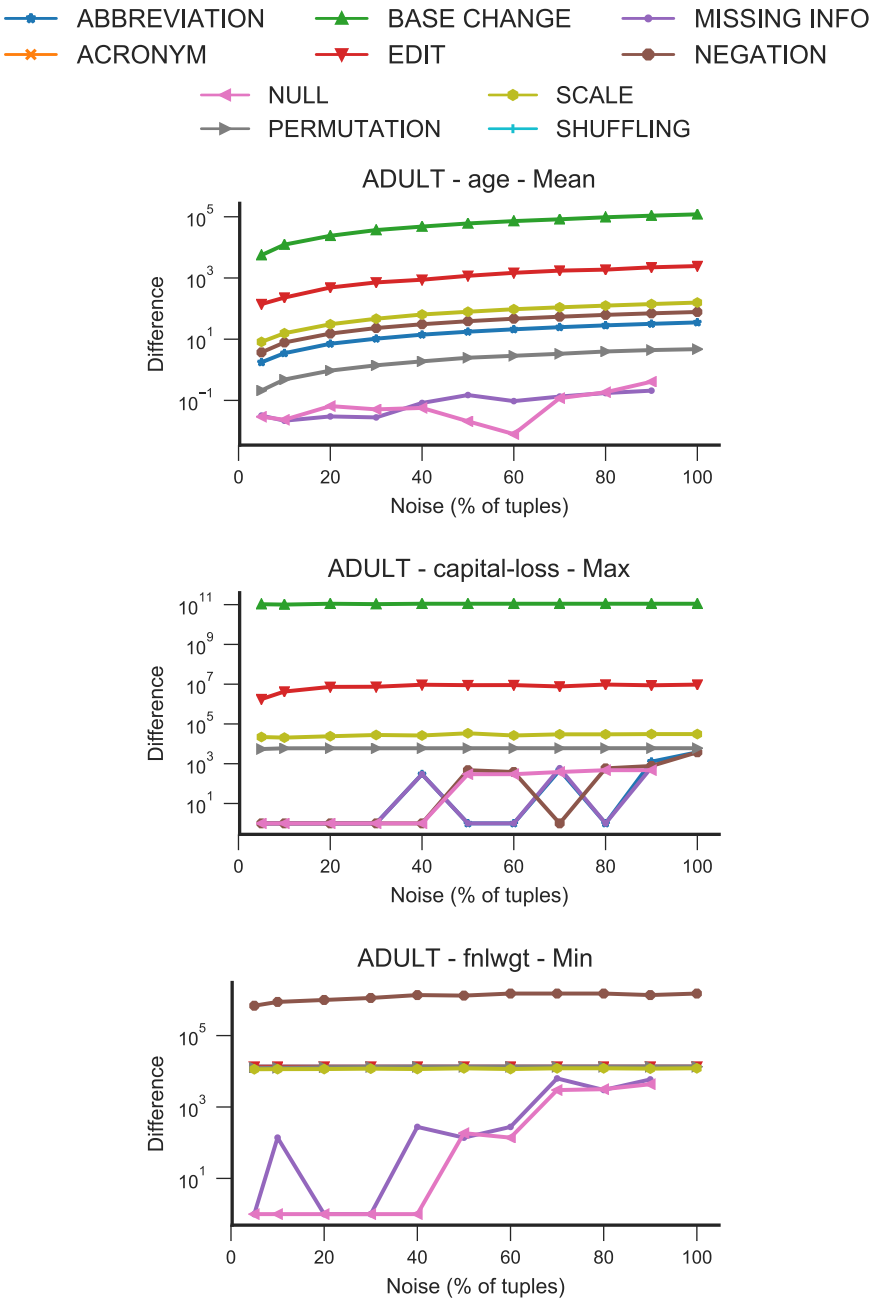

Figure B.1: `ADULT` dataset: mean of `age` column (top), max of `capital-loss` (middle), and min of `fnlwgt` (bottom).

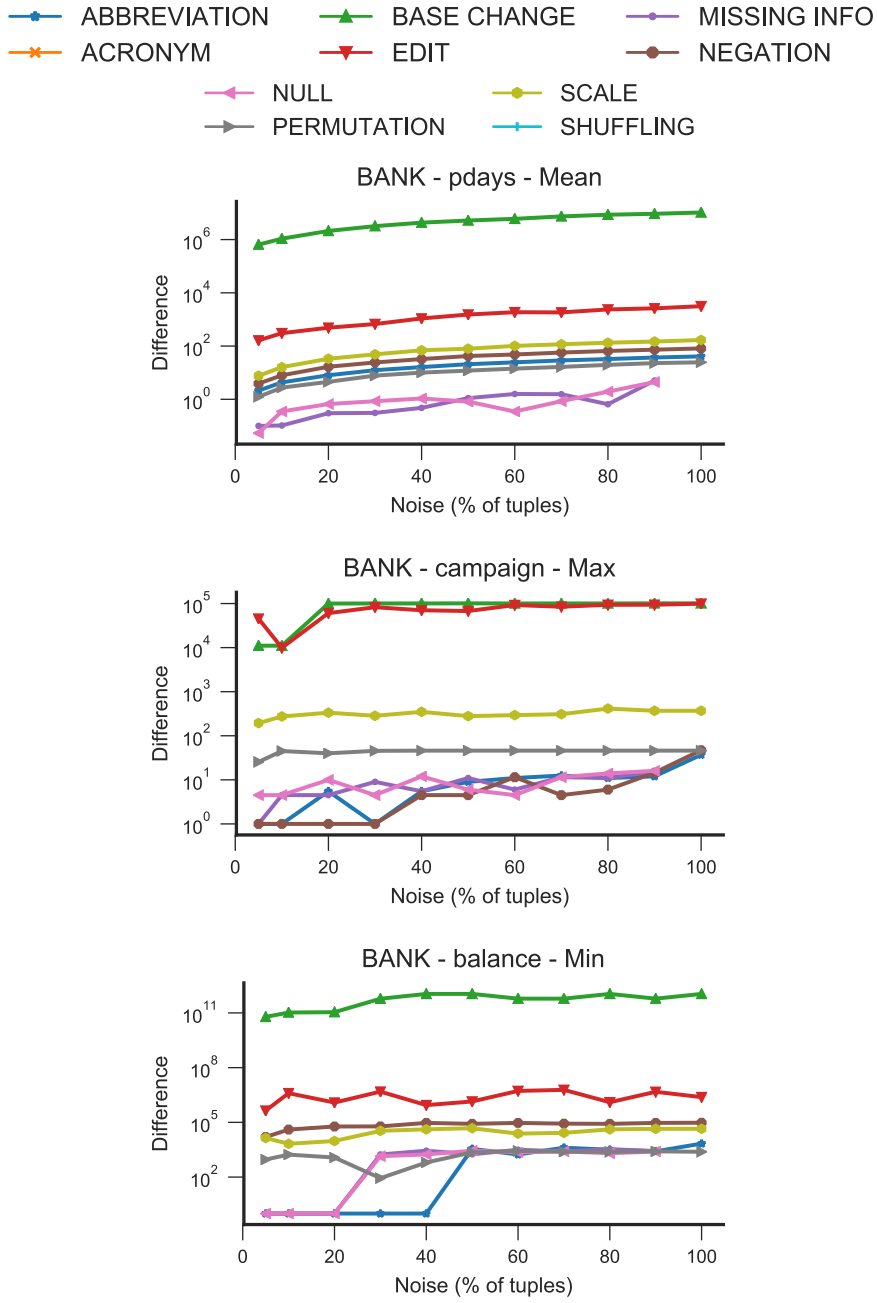

Figure B.2: `BANK` dataset: mean of `pdays` (top), max of `campaign` (middle), and min of `balance` (bottom).

Figure B.1 and Figure B.2 show that the analysis of the aggregations conducted reports similar results for `ADULT` and `BANK`. The 3 aggregations result in an increment in the distance while adding more noise into the data. `MISSING` and `NULL` do not alter much the results of the computation of the mean (first plot of each figure) since removing some values do not affect considerably the aggregation. For what concern the maximum (middle plot) and minimum (bottom plot), we have that there are some changes in the results. This is due to the randomness of the generation of the noise. It may happen that the actual maximum (minimum) value is affected by the noise, and hence it becomes a non-maximal (minimal) value, or that value is not changed. The other errors are more effective: `BASE CHANGE` and `EDIT` change the results of a factor up to 10^6 from the mean obtained with the original dataset.

C GENERIC DISTANCE

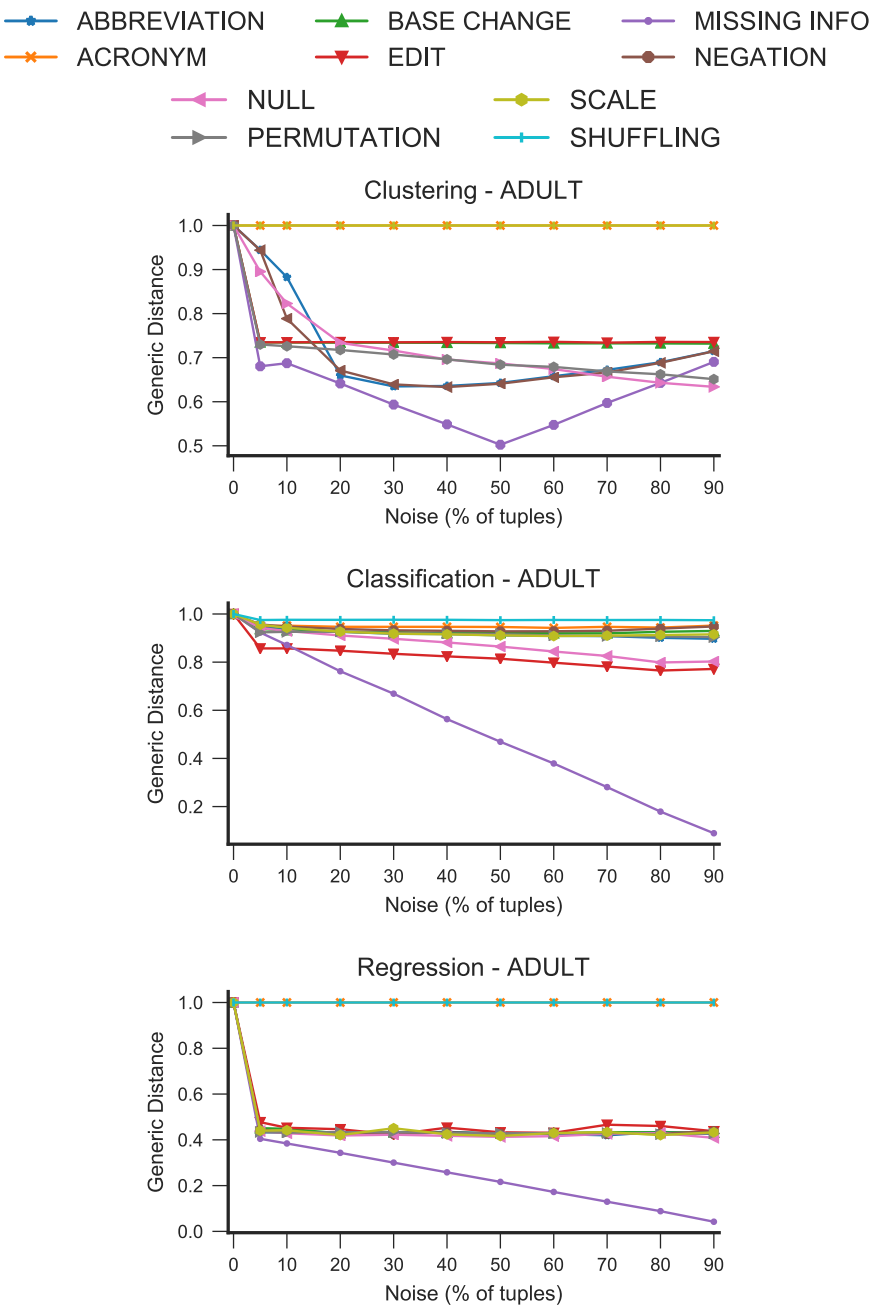

Figure C.1: Generic distance over the `ADULT` dataset, for clustering (top plot), classification (middle) and regression (bottom).

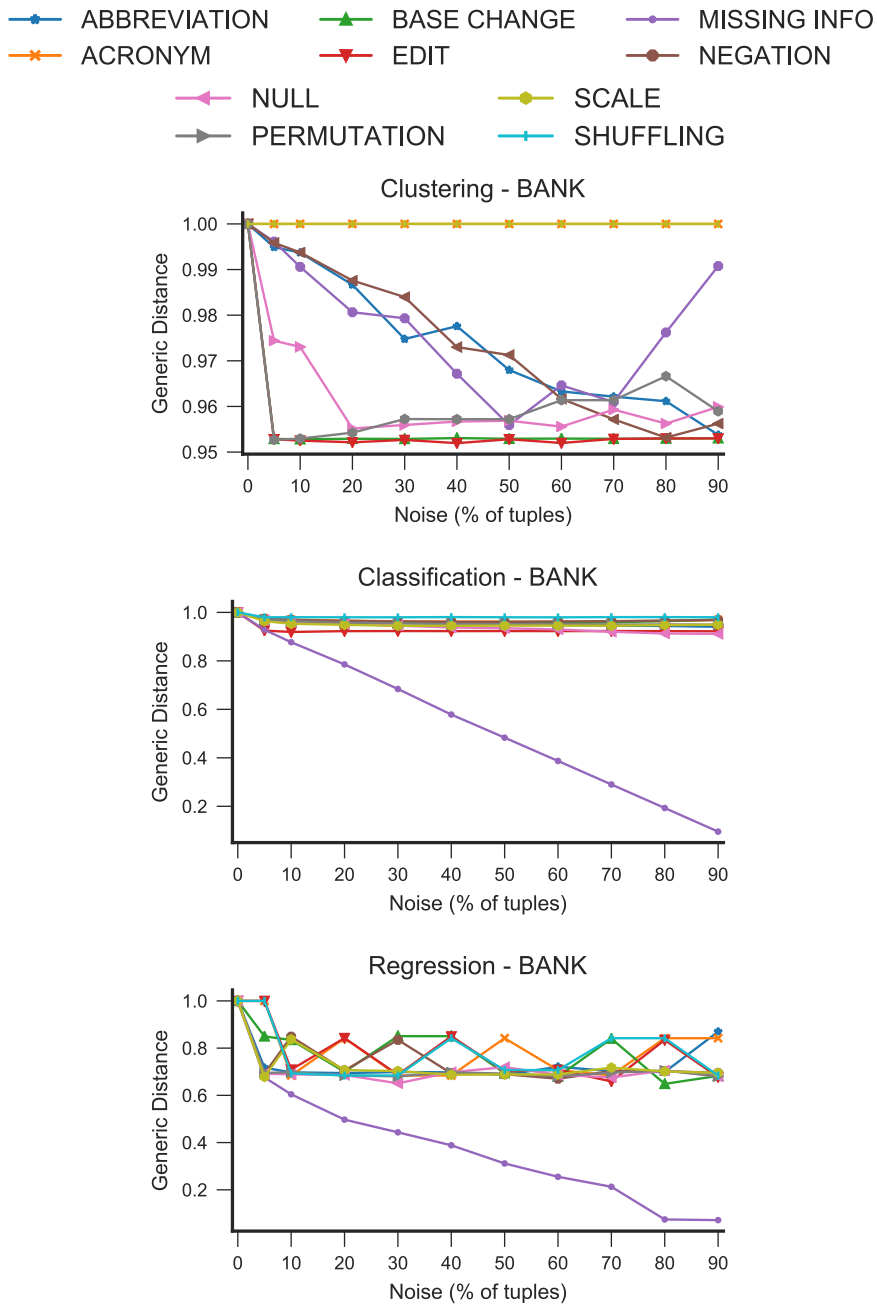

Figure C.2: Generic distance over the `BANK` dataset, for clustering (top plot), classification (middle) and regression (bottom).

Figure C.1 and Figure C.2 present the results for the ADULT and the BANK dataset, while Table C.1 reports the sensitivity factor evaluated over our custom generic distance. In the ADULT dataset, we see that NEGATION follows precisely the same parabolic behavior and ABBREVIATION has the final peak (not shown in the figure) found with the Fowlkes-Mallows score. Due to that pick, the ABBREVIATION lines are both categorized as irregular, while the NEGATION lines are both polynomial, with a slope of -1.08 in FM and -1.54 in the generic distance.

A similar behavior can be observed also for clustering on BANK, where ABBREVIATION presents the same peak with a high noise percentage (both distances categorize the noise as irregular). All the other noises are categorized by both distances as constants, meaning that the effect on the results do not change when increasing the percentage of errors. We note a similar situation in the AIRLINES dataset with NEGATION (polynomial slope of -1.92 for the task-specific distance vs. -1.89 of the generic) and NULL (polynomial slope of -1.57 vs. -0.93 of the generic distance), whereas EDIT and BASE CHANGE follow an irregular behavior as it was shown by the task-specific metric. Similar conclusion can be observed for classification, where the impact of the noises is higher in ADULT than in BANK. NULL, for example, has a linear relationship with a slope of -0.17 for the task-specific distance in the ADULT dataset, compared to -0.16 obtained with our generic distance, while in the BANK dataset is categorized as constants by both. To conclude, the custom distance in more than 63% of the times leads to the same results of the task specific one. Moreover, for classification this percentage reaches his peak with $\sim$ 97% of correct results.

Generic Distance		Clustering			Classification				Regression			
Noise		**Linear**	**Polynomial**	ρ	**Class**	**Linear**	**Polynomial**	ρ	**Class**	**Linear**	**Polynomial**	ρ	**Class**
ADULT	ABBREVIATION	(0.52, -0.30)	(0.70, -0.93)	**−0.48**	I	(0.59, -0.06)	(0.80, -0.20)	−0.90*	C	(0.19, -0.21)	(0.38, -0.96)	**−0.55**	I
	ACRONYM	(1.00, 0.00)	(1.00, 0.00)	0.00	C	(0.06, -0.01)	(0.55, -0.13)	−0.21	C	(1.00, 0.00)	(1.00, 0.00)	0.00	C
	BASE	(0.19, -0.10)	(0.37, -0.44)	−1.00*	C	(0.16, -0.03)	(0.63, -0.19)	−0.31	C	(0.20, -0.22)	(0.40, -0.98)	**−0.47**	I
	CHANGE												
	EDIT	(0.18, -0.09)	(0.37, -0.44)	**0.24**	I	(0.72, -0.16)	**(0.78, -0.33)**	−0.99*	P	(0.19, -0.20)	(0.41, -0.98)	**−0.22**	I
	MISSING INFO	–	–	–	–	**(1.00, -0.99)**	(1.00, -1.08)	−1.00*	L	(0.68, -0.63)	**(0.75, -1.37)**	−1.00*	P
	NEGATION	(0.00, 0.00)	**(0.83, -1.54)**	−0.03	P	(0.04, -0.01)	(0.84, -0.22)	−0.17	C	(0.00, 0.03)	(0.54, -1.67)	**−0.04**	I
	NULL	(0.29, -0.19)	**(0.84, -1.14)**	−0.27	P	**(0.92, -0.18)**	(0.96, -0.30)	−0.97*	L	(0.21, -0.22)	(0.39, -0.97)	**−0.69***	I
	PERMUTATION	**(0.76, -0.30)**	(0.91, -0.77)	−1.00*	L	(0.47, -0.05)	(0.59, -0.15)	−0.97*	C	(0.20, -0.21)	(0.38, -0.94)	−0.94*	C
	SCALE	(0.47, -0.19)	(0.59, -0.53)	−1.00*	C	(0.50, -0.06)	(0.85, -0.22)	−0.72*	C	(0.20, -0.22)	(0.39, -0.97)	**−0.55**	I
	SHUFFLING	(1.00, 0.00)	(1.00, 0.00)	0.00	C	(0.22, -0.01)	(0.38, -0.04)	−0.62*	C	(1.00, 0.00)	(1.00, 0.00)	0.00	C
BANK	ABBREVIATION	(0.40, -0.16)	(0.62, 0.25)	−0.99*	C	(0.79, -0.05)	(0.89, -0.10)	−1.00*	C	(0.03, -0.05)	(0.31, -0.57)	**0.11**	I
	ACRONYM	(1.00, 0.00)	(1.00, 0.00)	0.00	C	(0.10, -0.01)	(0.75, -0.10)	−0.24	C	(0.05, -0.08)	(0.46, -0.86)	**−0.25**	I
	BASE	(0.18, -0.02)	(0.36, -0.08)	0.32	C	(0.02, -0.00)	(0.76, -0.12)	−0.03	C	(0.46, -0.21)	(0.51, -0.46)	**−0.63***	I
	CHANGE												
	EDIT	(0.17, -0.02)	(0.37, -0.08)	0.27	C	(0.19, -0.03)	(0.36, -0.13)	−0.50	C	(0.40, -0.23)	(0.50, -0.65)	**−0.74***	I
	MISSING INFO	–	–	–	–	**(1.00, -0.99)**	(1.00, -1.03)	−1.00*	L	**(0.90, -0.80)**	(0.93, -1.35)	−1.00*	L
	NEGATION	(0.04, -0.01)	(0.91, -0.16)	−0.23	C	(0.07, -0.01)	(0.73, -0.10)	−0.16	C	(0.36, -0.18)	(0.48, -0.53)	**−0.53**	I
	NULL	(0.96, -0.05)	(0.98, -0.08)	−0.99*	C	(0.82, -0.07)	(0.92, -0.16)	−0.96*	C	(0.03, -0.05)	(0.42, -0.68)	**−0.08**	I
	PERMUTATION	(0.33, -0.02)	(0.74, -0.11)	−0.26	C	(0.53, -0.03)	(0.76, -0.10)	−0.99*	C	(0.15, -0.10)	(0.38, -0.55)	**−0.06**	I
	SCALE	(0.00, -0.00)	(0.28, -0.07)	0.47	C	(0.24, -0.02)	(0.67, -0.13)	−0.27	C	(0.24, -0.13)	(0.47, -0.60)	**−0.10**	I
	SHUFFLING	(1.00, 0.00)	(1.00, 0.00)	0.00	C	(0.17, -0.01)	(0.34, -0.03)	0.05	C	(0.18, -0.15)	(0.26, -0.50)	**−0.51**	I
AIRLINES	ABBREVIATION	**(0.87, -0.54)**	(0.91, -0.99)	−1.00*	L	(0.47, -0.32)	**(0.89, -1.43)**	−0.70*	P	(0.16, -0.18)	(0.35, -0.88)	**0.50**	I
	ACRONYM	(1.00, 0.00)	(1.00, 0.00)	0.00	C	(0.04, -0.00)	(0.45, -0.06)	−0.17	C	(1.00, 0.00)	(1.00, 0.00)	0.00	C
	BASE	(0.17, -0.23)	(0.36, -1.08)	**0.44**	I	(0.01, -0.04)	**(0.99, -1.45)**	−0.14	P	(0.18, -0.19)	(0.36, -0.88)	**0.16**	I
	CHANGE												
	EDIT	(0.17, 0.12)	(0.36, 0.57)	**0.04**	I	**(0.89, -0.70)**	(0.99, -1.56)	−0.99*	L	(0.24, -0.22)	(0.45, -0.93)	**−0.89***	I
	MISSING INFO	–	–	–	–	**(1.00, -1.02)**	(1.00, -1.23)	−1.00*	L	(0.72, -0.65)	**(0.78, -1.34)**	−1.00*	P
	NEGATION	(0.00, -0.03)	**(0.89, -1.89)**	−0.14	P	(0.02, -0.04)	**(0.99, -1.23)**	−0.19	P	(0.00, 0.03)	(0.52, -1.53)	**−0.14**	I
	NULL	**(0.88, -0.57)**	(0.91, -0.93)	−0.99*	L	**(0.94, -0.74)**	(1.00, -1.39)	−1.00*	L	(0.12, -0.15)	(0.38, -0.97)	**0.48**	I
	PERMUTATION	(0.00, 0.01)	(0.03, 0.26)	**−0.34**	I	**(0.93, -0.57)**	(1.00, -1.14)	−1.00*	L	(0.18, -0.19)	(0.37, -0.88)	**0.00**	I
	SCALE	(0.08, -0.16)	(0.54, -1.50)	**−0.04**	I	(0.62, -0.45)	**(1.00, -1.72)**	−0.70*	P	(0.19, -0.20)	(0.38, -0.89)	**−0.71***	I
	SHUFFLING	(1.00, 0.00)	(1.00, 0.00)	0.00	C	(0.14, -0.00)	(0.32, -0.02)	−0.19	C	(1.00, 0.00)	(1.00, 0.00)	0.00	C

Table C.1: 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). Each value is measured with the generic distance we implemented.

BIBLIOGRAPHY

[Abe+16a] Z. Abedjan et al. "Detecting Data Errors: Where are we and what needs to be done?" In: *PVLDB* 9.12 (2016) (cit. on pp. 6, 38).

[Abe+16b] Ziawasch Abedjan et al. "Detecting Data Errors: Where are we and what needs to be done?" In: *PVLDB* (2016), p. 993 (cit. on pp. 4, 5).

[Abe+18] Z. Abedjan et al. *Data Profiling*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2018. ISBN: 9781681734477. URL: https://books.google.it/books?id=LEF7DwAAQBAJ (cit. on p. 112).

[Abo+99] Gregory D Abowd et al. "Towards a better understanding of context and context-awareness". In: *Handheld and ubiquitous computing*. Springer. 1999, pp. 304–307 (cit. on p. 26).

[AGN15] Ziawasch Abedjan, Lukasz Golab, and Felix Naumann. "Profiling relational data: a survey". In: *VLDB J.* 24.4 (2015), pp. 557–581 (cit. on pp. 2, 3, 37).

[Apa19a] Apache Flink Team. *Apache Flink*. 2019. URL: http://flink.apache.org (cit. on p. 39).

[Apa19b] Apache Heron Team. *Apache Heron*. 2019. URL: http://heronstreaming.io (cit. on pp. 7, 39).

[Apa19c] Apache Lucene Team. *Apache Lucene*. 2019. URL: http://lucene.apache.org (cit. on p. 92).

[Apa19d] Apache Spark Team. *Apache Spark*. 2019. URL: http://spark.apache.org/streaming/ (cit. on p. 39).

[Apa19e] Apache Storm Team. *Apache Storm*. 2019. URL: http://storm.apache.org (cit. on p. 39).

[ARC16] Cristian Axenie, Christoph Richter, and Jörg Conradt. "A Self-Synthesis Approach to Perceptual Learning for Multisensory Fusion in Robotics". In: *Sensors* 16.10 (2016), p. 1751 (cit. on p. 96).

[Aro+15] Patricia C Arocena et al. "Messing up with BART: error generation for evaluating data-cleaning algorithms". In: *VLDB* 9.2 (2015) (cit. on pp. 6, 57).

[ASN14] Ziawasch Abedjan, Patrick Schulze, and Felix Naumann. "DFD: Efficient Functional Dependency Discovery". In: *CIKM*. 2014, pp. 949–958 (cit. on p. 37).

[ATV08] Bogdan Alexe, Wang Chiew Tan, and Yannis Velegrakis. "STBenchmark: towards a benchmark for mapping systems". In: *PVLDB* 1.1 (2008), p. 230 (cit. on p. 57).

[Bat+08] Carlo Batini et al. "A Comprehensive Data Quality Methodology for Web and Structured Data". In: *IJICA*. Vol. 1. 3. 2008, pp. 205–218 (cit. on pp. 2, 3, 36).

[Bat+09] Carlo Batini et al. "Methodologies for data quality assessment and improvement". In: *CSUR* 41.3 (2009), p. 16 (cit. on pp. 2, 3, 13, 32, 35).

[Bat+15] Carlo Batini et al. "From data quality to big data quality". In: *JDM* 26.1 (2015), pp. 60–82 (cit. on pp. 1, 2, 37, 38, 66).

[Bau+07] Jana Bauckmann et al. "Efficiently Detecting Inclusion Dependencies". In: *ICDE*. 2007, pp. 1448–1450 (cit. on p. 23).

[Bau+12] Jana Bauckmann et al. "Discovering conditional inclusion dependencies". In: *CIKM*. 2012, pp. 2094–2098 (cit. on pp. 18, 23, 37).

[BC04] Leopoldo Bertossi and Jan Chomicki. "Query answering in inconsistent databases". In: *Logics for emerging applications of databases*. Springer, 2004, pp. 43–83 (cit. on pp. 38, 52).

[Ben+09] Omar Benjelloun et al. "Swoosh: a generic approach to entity resolution". In: *VLDBJ* 18.1 (2009), pp. 255–276 (cit. on p. 43).

[Ber+15] Moria Bergman et al. "Query-Oriented Data Cleaning with Oracles". In: *SIGMOD*. 2015, pp. 1199–1214 (cit. on pp. 3, 21, 22, 38).

[Ber11] Leopoldo E. Bertossi. *Database Repairing and Consistent Query Answering*. Morgan & Claypool Publishers, 2011 (cit. on p. 14).

[BG07] Indrajit Bhattacharya and Lise Getoor. "Collective entity resolution in relational data". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1.1 (2007), p. 5 (cit. on p. 45).

[Boh+05] Philip Bohannon et al. "A cost-based model and effective heuristic for repairing constraints by value modification". In: *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. ACM. 2005, pp. 143–154 (cit. on pp. 16, 17).

[Bol+07a] Cristiana Bolchini et al. "A data-oriented survey of context models". In: *ACM Sigmod Record* 36.4 (2007), pp. 19–26 (cit. on p. 26).

[Bol+07b] Cristiana Bolchini et al. "Using context for the extraction of relational views". In: *Modeling and Using Context*. Springer, 2007, pp. 108–121 (cit. on p. 27).

[Bol+09] Cristiana Bolchini et al. "And what can context do for data?" In: *Communications of the ACM* 52.11 (2009), pp. 136–140 (cit. on pp. 26, 27, 31).

[BP85] Donald P Ballou and Harold L Pazer. "Modeling data and process quality in multi-input, multi-output information systems". In: *Management science* 31.2 (1985), pp. 150–162 (cit. on p. 16).

[Bre01] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32 (cit. on p. 63).

[BRJ11] Leopoldo Bertossi, Flavio Rizzolo, and Lei Jiang. "Data quality is context dependent". In: *Enabling Real-Time Business Intelligence*. Springer, 2011, pp. 52–67 (cit. on p. 30).

[BS06] Carlo Batini and Monica Scannapieco. *Data Quality: Concepts, Methodologies and Techniques*. Springer, 2006 (cit. on pp. 3, 4, 15, 16, 36).

[BSB10] Daniele Barone, Fabio Stella, and Carlo Batini. "Dependency discovery in data quality". In: *Advanced Information Systems Engineering*. Springer. 2010, pp. 53–67 (cit. on p. 30).

[BSM03] Matthew Bovee, Rajendra P Srivastava, and Brenda Mak. "A conceptual framework and belief-function approach to assessing overall information quality". In: *International journal of intelligent systems* 18.1 (2003), pp. 51–74 (cit. on pp. 20, 22).

[BST04] Cristiana Bolchini, Fabio A Schreiber, and Letizia Tanca. "A context-aware methodology for very small data base design". In: *ACM SIGMOD Record* 33.1 (2004), pp. 71–76 (cit. on pp. 3, 26).

[Cap+18] Cinzia Cappiello et al. "Validating Data Quality Actions in Scoring Processes". In: *JDIQ* 9.2 (2018), p. 11 (cit. on p. 38).

[Cas13] Federico Castanedo. "A review of data fusion techniques". In: *The Scientific World Journal* 2013 (2013) (cit. on p. 19).

[CFY13] Yang Cao, Wenfei Fan, and Wenyuan Yu. "Determining the relative accuracy of attributes". In: *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. ACM. 2013, pp. 565–576 (cit. on pp. 16, 19).

[CGY] Barbara Catania, Giovanna Guerrini, and Beyza Yaman. "Context-Dependent Quality-Aware Source Selection for Live Queries on Linked Data". In: *Update* 2 (), p. 5 (cit. on p. 31).

[Chi+16] Sanket Chintapalli et al. "Benchmarking Streaming Computation Engines: Storm, Flink and Spark Streaming". In: *IPDPS*. 2016, pp. 1789–1792 (cit. on p. 39).

[Chr+19] Vassilis Christophides et al. "End-to-End Entity Resolution for Big Data: A Survey". In: *arXiv preprint arXiv:1905.06397* (2019) (cit. on p. 42).

[Chu+15] Xu Chu et al. "KATARA: A Data Cleaning System Powered by Knowledge Bases and Crowdsourcing". In: *SIGMOD*. 2015, pp. 1247–1261 (cit. on pp. 3, 16, 38).

[CM08] Fei Chiang and Renée J. Miller. "Discovering data quality rules". In: *PVLDB* 1.1 (2008), pp. 1166–1177 (cit. on pp. 3, 4, 16, 18).

[Coc+01] Munir Cochinwala et al. "Efficient data reconciliation". In: *Information Sciences* 137.1-4 (2001), pp. 1–15 (cit. on p. 43).

[Con+07] Gao Cong et al. "Improving Data Quality: Consistency and Accuracy". In: *PVLDB*. 2007, pp. 315–326 (cit. on pp. 3, 4, 16, 18, 23, 37).

[Cou+05] Joëlle Coutaz et al. "Context is key". In: *Communications of the ACM* 48.3 (2005), pp. 49–53 (cit. on p. 27).

[CT05] Jan Chomicki and David Toman. "Temporal databases". In: *Foundations of Artificial Intelligence* 1 (2005), pp. 429–467 (cit. on p. 23).

[CZ15] Li Cai and Yangyong Zhu. "The challenges of data quality and data quality assessment in the big data era". In: *DSJ* 14 (2015) (cit. on p. 37).

[DBB06] Fabrizio De Amicis, Daniele Barone, and Carlo Batini. "An Analytical Framework to Analyze Dependencies Among Data Quality Dimensions." In: *ICIQ*. 2006, pp. 369–383 (cit. on pp. 28, 29).

[DN11] Uwe Draisbach and Felix Naumann. "A generalization of blocking and windowing algorithms for duplicate detection". In: *2011 International Conference on Data and Knowledge Engineering (ICDKE)*. IEEE. 2011, pp. 18–24 (cit. on p. 44).

[Eck02] Wayne W Eckerson. "Data quality and the bottom line: Achieving business success through a commitment to high quality data". In: *The Data Warehousing Institute* (2002), pp. 1–36 (cit. on p. 1).

[EIV06] Ahmed K Elmagarmid, Panagiotis G Ipeirotis, and Vassilios S Verykios. "Duplicate record detection: A survey". In: *IEEE Transactions on knowledge and data engineering* 19.1 (2006), pp. 1–16 (cit. on p. 43).

[EIV07] Ahmed K Elmagarmid, Panagiotis G Ipeirotis, and Vassilios S Verykios. "Duplicate record detection: A survey". In: *Knowledge and Data Engineering, IEEE Transactions on* 19.1 (2007), pp. 1–16 (cit. on p. 24).

[Fan+08] Wenfei Fan et al. "Conditional functional dependencies for capturing data inconsistencies". In: *ACM Transactions on Database Systems (TODS)* 33.2 (2008), p. 6 (cit. on pp. 16, 18).

[Fan+09] Wenfei Fan et al. "Reasoning about record matching rules". In: *PVLDB* 2.1 (2009), pp. 407–418 (cit. on p. 42).

[Fan+11] Wenfei Fan et al. "Discovering conditional functional dependencies". In: *Knowledge and Data Engineering, IEEE Transactions on* 23.5 (2011), pp. 683–698 (cit. on pp. 16, 18).

[Fan+14] Wenfei Fan et al. "Conflict resolution with data currency and consistency". In: *Journal of Data and Information Quality (JDIQ)* 5.1-2 (2014), p. 6 (cit. on p. 23).

[Fan15] Wenfei Fan. "Data Quality: From Theory to Practice". In: *SIGMOD Rec.* 44.3 (Dec. 2015), pp. 7–18 (cit. on pp. 14, 15).

[FG10a] Wenfei Fan and Floris Geerts. "Capturing missing tuples and missing values". In: *PODS*. 2010, pp. 169–178 (cit. on pp. 3, 37).

[FG10b] Wenfei Fan and Floris Geerts. "Capturing missing tuples and missing values". In: *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM. 2010, pp. 169–178 (cit. on p. 21).

[FG10c] Wenfei Fan and Floris Geerts. "Relative information completeness". In: *ACM Transactions on Database Systems (TODS)* 35.4 (2010), p. 27 (cit. on p. 21).

[FG12] Wenfei Fan and Floris Geerts. *Foundations of Data Quality Management*. Morgan & Claypool Publishers, 2012 (cit. on pp. 1, 13–15, 23, 24).

[FGJ09] Wenfei Fan, Floris Geerts, and Xibei Jia. "Conditional dependencies: A principled approach to improving data quality". In: *Dataspace: The Final Frontier*. Springer, 2009, pp. 8–20 (cit. on pp. 16, 18).

[FGW11] Wenfei Fan, Floris Geerts, and Jef Wijsen. "Determining the currency of data". In: *PODS*. 2011, pp. 71–82 (cit. on pp. 22, 23, 37).

[Flo+17] Avrilia Floratou et al. "Dhalion: Self-Regulating Stream Processing in Heron". In: *PVLDB* 10.12 (2017), pp. 1825–1836 (cit. on pp. 2, 7, 39).

[FM83] Edward B Fowlkes and Colin L Mallows. "A method for comparing two hierarchical clusterings". In: *JASA* 78.383 (1983), pp. 553, 569 (cit. on pp. 5, 59, 73).

[FS69] Ivan P Fellegi and Alan B Sunter. "A theory for record linkage". In: *Journal of the American Statistical Association* 64.328 (1969), pp. 1183–1210 (cit. on pp. 43, 44).

[Fu+15] Tom Z. J. Fu et al. "DRS: Dynamic Resource Scheduling for Real-Time Analytics over Fast Streams". In: *ICDCS*. 2015, pp. 411–420 (cit. on p. 41).

[Gia+10] Fosca Giannotti et al. "Mobility data mining: discovering movement patterns from trajectory data". In: *Proceedings of the Second International Workshop on Computational Transportation Science, San Jose, CA, USA, November 2, 2010. Proceedings*. 2010, pp. 7–10 (cit. on p. 2).

[GKS11] Lukasz Golab, Flip Korn, and Divesh Srivastava. "Efficient and Effective Analysis of Data Quality using Pattern Tableaux." In: *IEEE Data Eng. Bull.* 34.3 (2011), pp. 26–33 (cit. on p. 2).

[GM13] Stephen C Guptill and Joel L Morrison. *Elements of spatial data quality*. Elsevier, 2013 (cit. on p. 25).

[Goo80] Michael F Goodchild. "Fractals and the accuracy of geographical measures". In: *Journal of the International Association for Mathematical Geology* 12.2 (1980), pp. 85–98 (cit. on p. 25).

[Hal01] Alon Y Halevy. "Answering queries using views: A survey". In: *The VLDB Journal* 10.4 (2001), pp. 270–294 (cit. on p. 30).

[Han+14] Zheng Han et al. "Elastic Allocator: An Adaptive Task Scheduler for Streaming Query in the Cloud". In: *SOSE*. 2014, pp. 284–289 (cit. on pp. 2, 7, 41).

[Her+11] Herodotos Herodotou et al. "Starfish: A Self-tuning System for Big Data Analytics". In: *CIDR*. 2011, pp. 261–272 (cit. on p. 40).

[Hin+11] Benjamin Hindman et al. "Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center". In: *USENIX NSDI*. 2011 (cit. on p. 39).

[HS95] Mauricio A Hernández and Salvatore J Stolfo. "The merge/purge problem for large databases". In: *ACM Sigmod Record*. Vol. 24. 2. ACM. 1995, pp. 127–138 (cit. on p. 44).

[IC+15] Ihab F Ilyas, Xu Chu, et al. "Trends in cleaning relational data: Consistency and deduplication". In: *Foundations and Trends® in Databases* 5.4 (2015), pp. 281–393 (cit. on p. 1).

[IPC15] Stratos Idreos, Olga Papaemmanouil, and Surajit Chaudhuri. "Overview of data exploration techniques". In: *SIGMOD*. ACM. 2015, pp. 277–281 (cit. on p. 66).

[Jar+13] Matthias Jarke et al. *Fundamentals of data warehouses*. Springer Science & Business Media, 2013 (cit. on p. 20).

[JGP15] Manas Joglekar, Hector Garcia-Molina, and Aditya G. Parameswaran. "Smart Drill-Down: A New Data Exploration Operator". In: *PVLDB* 8.12 (2015), pp. 1928–1931. DOI: 10.14778/2824032.2824103. URL: http://www.vldb.org/pvldb/vol8/p1928-joglekar.pdf (cit. on p. 112).

[KBI18] Shrinu Kushagra, Shai Ben-David, and Ihab Ilyas. "Semi-supervised clustering for de-duplication". In: *arXiv preprint arXiv:1810.04361* (2018) (cit. on p. 9).

[Kho+17] Alireza Khoshkbarforoushha et al. "Flower: A Data Analytics Flow Elasticity Manager". In: *PVLDB* 10.12 (2017), pp. 1893–1896 (cit. on p. 41).

[KKR17] Alireza Khoshkbarforoushha, Alireza Khosravian, and Rajiv Ranjan. "Elasticity management of Streaming Data Analytics Flows on clouds". In: *JCSS* 89 (2017), pp. 24–40 (cit. on p. 41).

[Kon+16] Pradap Konda et al. "Magellan: Toward Building Entity Matching Management Systems over Data Science Stacks". In: *PVLDB* 9.13 (2016), pp. 1581–1584 (cit. on pp. 46, 113, 115).

[KPN15] Sebastian Kruse, Paolo Papotti, and Felix Naumann. "Estimating Data Integration and Cleaning Effort." In: *EDBT*. 2015, pp. 61–72 (cit. on p. 4).

[KR10] Hanna Köpcke and Erhard Rahm. "Frameworks for entity matching: A comparison". In: *Data & Knowledge Engineering* 69.2 (2010), pp. 197–210 (cit. on p. 42).

[Kri+16] Sanjay Krishnan et al. "ActiveClean: interactive data cleaning for statistical modeling". In: *PVLDB* 9.12 (2016), pp. 948–959 (cit. on pp. 6, 38).

[Kri+17] S. Krishnan et al. "BoostClean: Automated Error Detection and Repair for Machine Learning". In: *CoRR* (2017) (cit. on p. 6).

[KS98] Beverly K Kahn and Diane M Strong. "Product and Service Performance Model for Information Quality: An Update." In: *IQ*. 1998, pp. 102–115 (cit. on p. 35).

[KTR10] Hanna Köpcke, Andreas Thor, and Erhard Rahm. "Evaluation of entity resolution approaches on real-world match problems". In: *Proceedings of the VLDB Endowment* 3.1-2 (2010), pp. 484–493 (cit. on p. 24).

[Kuh55] Harold W Kuhn. "The Hungarian method for the assignment problem". In: *NRL* 2.1-2 (1955), pp. 83–97 (cit. on p. 5).

[KW03a] Henryk Krawczyk and Bogdan Wiszniewski. "Digital document life cycle development". In: *Proceedings of the 1st international symposium on Information and communication technologies*. Trinity College Dublin. 2003, pp. 255–260 (cit. on p. 25).

[KW03b] Henryk Krawczyk and Bogdan Wiszniewski. "Visual GQM approach to quality-driven development of electronic documents". In: *Proceedings of the Second International Workshop on Web Document Analysis*. 2003, pp. 43–46 (cit. on p. 25).

[LC02] Liping Liu and Lauren Chi. "Evolutional Data Quality: A Theory-Specific View." In: *IQ*. 2002, pp. 292–304 (cit. on p. 26).

[Lee+02] Yang W. Lee et al. "AIMQ: a methodology for information quality assessment". In: *Information & Management* 40.2 (2002), pp. 133–146 (cit. on pp. 2, 15, 35).

[Lee+09] Yang W Lee et al. *Journey to data quality*. The MIT Press, 2009 (cit. on p. 13).

[Li+12] Xian Li et al. "Truth finding on the deep web: is the problem solved?" In: *Proceedings of the VLDB Endowment* 6.2 (2012), pp. 97–108 (cit. on p. 19).

[Lin91] Jianhua Lin. "Divergence measures based on the Shannon entropy". In: *Information Theory, IEEE Transactions on* 37.1 (1991), pp. 145–151 (cit. on p. 29).

[Liv+19] Ester Livshits et al. "Principles of Progress Indicators for Database Repairing". In: *arXiv preprint arXiv:1904.06492* (2019) (cit. on p. 4).

[LN16] Philipp Langer and Felix Naumann. "Efficient order dependency detection". In: *VLDB J.* 25.2 (2016), pp. 223–241 (cit. on p. 63).

[LÖ09] Ling Liu and M. Tamer Özsu, eds. *Encyclopedia of Database Systems*. Springer, 2009 (cit. on p. 24).

[Lom+17] Federico Lombardi et al. "Elastic symbiotic scaling of operators and resources in stream processing systems". In: *IEEE Transactions on Parallel and Distributed Systems* 29.3 (2017), pp. 572–585 (cit. on p. 39).

[LRU14] Jure Leskovec, Anand Rajaraman, and Jeffrey D. Ullman. *Mining of Massive Datasets, 2nd Ed*. Cambridge University Press, 2014 (cit. on p. 2).

[LSB15] Nuno Laranjeiro, Seyma Nur Soydemir, and Jorge Bernardino. "A Survey on Data Quality: Classifying Poor Data". In: *Dependable Computing (PRDC), 2015 IEEE 21st Pacific Rim International Symposium on*. IEEE. 2015, pp. 179–188 (cit. on pp. 13, 24).

[MA19] Mohammad Mahdavi and Ziawasch Abedjan. "REDS: Estimating the Performance of Error Detection Strategies Based on Dirtiness Profiles". In: *SSDBM*. 2019, pp. 193–196 (cit. on p. 2).

[Mah+19] Mohammad Mahdavi et al. "Raha: A configuration-free error detection system". In: *Proceedings of the 2019 International Conference on Management of Data*. ACM. 2019, pp. 865–882 (cit. on pp. 47, 116).

[Mer+16] Jorge Merino et al. "A data quality in use model for big data". In: *FGCS* 63 (2016), pp. 123–130 (cit. on p. 38).

[MLC11] Sergio Moro, Raul Laureano, and Paulo Cortez. "Using data mining for bank direct marketing: An application of the crisp-dm methodology". In: *ESM*. 2011, pp. 117–121 (cit. on p. 64).

[Mot+17] Davide Mottin et al. "New trends on exploratory methods for data analytics". In: *Proceedings of the VLDB Endowment* 10.12 (2017), pp. 1977–1980 (cit. on p. 66).

[Mud+18] Sidharth Mudgal et al. "Deep learning for entity matching: A design space exploration". In: *Proceedings of the 2018 International Conference on Management of Data*. ACM. 2018, pp. 19–34 (cit. on pp. 8, 115).

[Nau02] Felix Naumann. *Quality-driven query answering for integrated information systems*. Vol. 2261. Springer Science & Business Media, 2002 (cit. on pp. 20, 22).

[Nau13] Felix Naumann. "Data profiling revisited". In: *SIGMOD Rec.* 42.4 (2013), pp. 40–49 (cit. on pp. 2, 68).

[Net+96] John Neter et al. *Applied linear statistical models*. Vol. 4. Irwin Chicago, 1996 (cit. on p. 61).

[OE16] Ziad Obermeyer and Ezekiel J Emanuel. "Predicting the future—big data, machine learning, and clinical medicine". In: *The New England journal of medicine* 375.13 (2016), p. 1216 (cit. on p. 1).

[Pap+15] Thorsten Papenbrock et al. "Data Profiling with Metanome". In: *PVLDB* 8.12 (2015), pp. 1860–1863 (cit. on p. 37).

[Pap+18] George Papadakis et al. "The return of jedAI: end-to-end entity resolution for structured and semi-structured data". In: *Proceedings of the VLDB Endowment* 11.12 (2018), pp. 1950–1953 (cit. on p. 46).

[Par+18] Noseong Park et al. "Data synthesis based on generative adversarial networks". In: *PVLDB* 11.10 (2018), pp. 1071–1083 (cit. on p. 64).

[PLW02] Leo Pipino, Yang W. Lee, and Richard Y. Wang. "Data quality assessment". In: *Commun. ACM* 45.4 (2002), pp. 211–218 (cit. on pp. 21, 35).

[Pow11] David Martin Powers. "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation". In: (2011) (cit. on p. 59).

[PPH16] Shelan Perera, Ashansa Perera, and Kamal Hakimzadeh. "Reproducible Experiments for Comparing Apache Flink and Apache Spark on Public Clouds". In: *CoRR* abs/1610.04493 (2016) (cit. on p. 39).

[Rat+17] Alexander Ratner et al. "Snorkel: Rapid training data creation with weak supervision". In: *VLDB* 11.3 (2017), pp. 269–282 (cit. on pp. 43, 57).

[RD00] Erhard Rahm and Hong Hai Do. "Data cleaning: Problems and current approaches". In: *IEEE Data Eng. Bull.* 23.4 (2000), pp. 3–13 (cit. on p. 24).

[RDG11] Vibhor Rastogi, Nilesh N. Dalvi, and Minos N. Garofalakis. "Large-Scale Collective Entity Matching". In: *PVLDB* 4.4 (2011), pp. 208–218 (cit. on p. 45).

[Red01] Thomas C Redman. *Data quality: the field guide*. Digital press, 2001 (cit. on p. 22).

[Red96] Thomas C. Redman. *Data quality for the information age*. Artech House, 1996 (cit. on p. 16).

[Rek+15] Theodoros Rekatsinas et al. "Finding quality in quantity: The challenge of discovering valuable sources for integration". In: *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*. 2015 (cit. on pp. 16, 19).

[Rus18] Gabriele Russo Russo. "Towards Decentralized Auto-Scaling Policies for Data Stream Processing Applications". In: *ZEUS*. 2018, pp. 47–54 (cit. on pp. 2, 7, 41).

[San+16] Donatello Santoro et al. "BART in action: Error generation and empirical evaluations of data-cleaning systems". In: *Proceedings of the 2016 International Conference on Management of Data*. ACM. 2016, pp. 2161–2164 (cit. on p. 6).

[Sch+09] Scott Schneider et al. "Elastic scaling of data parallel operators in stream processing". In: *2009 IEEE International Symposium on Parallel & Distributed Processing*. IEEE. 2009, pp. 1–12 (cit. on p. 40).

[Sch+18] Sebastian Schelter et al. "Automating large-scale data quality verification". In: *VLDB* 11.12 (2018), pp. 1781–1794 (cit. on p. 66).

[SFG03] Wenzhong Shi, Peter Fisher, and Michael F Goodchild. *Spatial data quality*. CRC Press, 2003 (cit. on p. 25).

[Sha51] Claude E Shannon. "Prediction and entropy of printed English". In: *Bell system technical journal* 30.1 (1951), pp. 50–64 (cit. on p. 29).

[Sin+17] Rohit Singh et al. "Synthesizing entity matching rules by examples". In: *Proceedings of the VLDB Endowment* 11.2 (2017), pp. 189–202 (cit. on p. 9).

[SLW97] Diane M. Strong, Yang W. Lee, and Richard Y. Wang. "Data Quality in Context". In: *Commun. ACM* 40.5 (1997), pp. 103–110 (cit. on pp. 15, 28).

[Spe04] Charles Spearman. "The proof and measurement of association between two things". In: *AJP* 15.1 (1904), pp. 72–101 (cit. on p. 62).

[SS14] Barna Saha and Divesh Srivastava. "Data quality: The other face of big data". In: *ICDE*. 2014, pp. 1294–1297 (cit. on pp. 1, 2, 12, 37).

[SS18] Anshu Shukla and Yogesh Simmhan. "Toward reliable and rapid elasticity for streaming dataflows on clouds". In: *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE. 2018, pp. 1096–1106 (cit. on p. 40).

[SW98] Claude E Shannon and Warren Weaver. *The mathematical theory of communication*. University of Illinois press, 1998 (cit. on p. 125).

[SWZ00] Ganesan Shankaranarayanan, Richard Y Wang, and Mostapha Ziad. "IP-MAP: Representing the Manufacture of an Information Product." In: *IQ*. 2000, pp. 1–16 (cit. on p. 34).

[TB98] Giri Kumar Tayi and Donald P. Ballou. "Examining Data Quality - Introduction". In: *Commun. ACM* 41.2 (1998), pp. 54–57 (cit. on p. 12).

[TPC19] TPC-H Team. *TPC-H*. 2019. URL: `http://www.tpc.org/tpch/` (cit. on p. 97).

[Van92] Ron Van Der Meyden. "The complexity of querying indefinite data about linearly ordered domains". In: *Proceedings of the eleventh ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*. ACM. 1992, pp. 331–345 (cit. on p. 21).

[Vav+13] Vinod Kumar Vavilapalli et al. "Apache Hadoop YARN: yet another resource negotiator". In: *SOCC*. 2013, 5:1–5:16 (cit. on p. 39).

[VEB10] Nguyen Xuan Vinh, Julien Epps, and James Bailey. "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance". In: *JMLR* 11.Oct (2010), pp. 2837–2854 (cit. on p. 59).

[Wan+11] Jiannan Wang et al. "Entity Matching: How Similar is Similar". In: *Proceedings of the VLDB Endowment* 4.10 (2011), pp. 622–633 (cit. on p. 46).

[Wan+14] Jiannan Wang et al. "A sample-and-clean framework for fast and accurate query processing on dirty data". In: *SIGMOD*. 2014, pp. 469–480 (cit. on p. 38).

[Wan98] Richard Y. Wang. "A Product Perspective on Total Data Quality Management". In: *Commun. ACM* 41.2 (1998), pp. 58–65 (cit. on pp. 15, 34).

[WBP13] Philip Woodall, Alexander Borek, and Ajith Kumar Parlikad. "Data quality assessment: the hybrid approach". In: *Information & management* 50.7 (2013), pp. 369–382 (cit. on p. 36).

[WC13] Guoping Wang and Chee-Yong Chan. "Multi-query optimization in mapreduce framework". In: *Proceedings of the VLDB Endowment* 7.3 (2013), pp. 145–156 (cit. on p. 46).

[WG13] Steven Euijong Whang and Hector Garcia-Molina. "Joint entity resolution on multiple datasets". In: *VLDB J.* 22.6 (2013), pp. 773–795 (cit. on p. 45).

[Wha+09] Steven Euijong Whang et al. "Entity resolution with iterative blocking". In: *SIGMOD* (2009), pp. 219–232 (cit. on pp. 42, 45).

[WM05] Cort J Willmott and Kenji Matsuura. "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance". In: *Climate research* 30.1 (2005), pp. 79, 82 (cit. on p. 59).

[WS96] Richard Y. Wang and Diane M. Strong. "Beyond Accuracy: What Data Quality Means to Data Consumers". In: *JMIS* 12.4 (1996), pp. 5–33 (cit. on pp. 1–4, 11, 12, 14, 15, 26).

[WSE09] Stephanie Watts, Ganesan Shankaranarayanan, and Adir Even. "Data quality assessment in context: A cognitive perspective". In: *Decision Support Systems* 48.1 (2009), pp. 202–211 (cit. on p. 32).

[WW96] Yair Wand and Richard Y Wang. "Anchoring data quality dimensions in ontological foundations". In: *Communications of the ACM* 39.11 (1996), pp. 86–95 (cit. on pp. 15, 20, 22).

[Yan+07] Su Yan et al. "Adaptive sorted neighborhood methods for efficient record linkage". In: *Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*. ACM. 2007, pp. 185–194 (cit. on p. 44).

[Zha+13] Zhenjie Zhang et al. "ABACUS: An Auction-Based Approach to Cloud Service Differentiation". In: *IC2E*. 2013, pp. 292–301 (cit. on p. 39).

[Zhu+18] Hui-Juan Zhu et al. "A Type-Based Blocking Technique for Efficient Entity Resolution over Large-Scale Data". In: *J. Sensors* 2018 (2018), 2094696:1–2094696:12 (cit. on pp. 2, 46).