Accepted Manuscript

Context-aware data quality assessment for big data

Danilo Ardagna, Cinzia Cappiello, Walter Samá, Monica Vitali

PII: S0167-739X(17)32915-1
DOI: https://doi.org/10.1016/j.future.2018.07.014
Reference: FUTURE 4332

To appear in: *Future Generation Computer Systems*

Received date : 20 December 2017
Revised date : 15 May 2018
Accepted date : 10 July 2018

Please cite this article as: D. Ardagna, C. Cappiello, W. Samá, M. Vitali, Context-aware data quality assessment for big data, *Future Generation Computer Systems* (2018), https://doi.org/10.1016/j.future.2018.07.014

Context-aware Data Quality Assessment for Big Data

Danilo Ardagna, Cinzia Cappiello, Walter Samá, Monica Vitali*

DEIB, Politecnico di Milano
Piazza Leonardo da Vinci 32, 20133 Milan, Italy

Abstract

Big data changed the way in which we collect and analyze data. In particular, the amount of available information is constantly growing and organizations rely more and more on data analysis in order to achieve their competitive advantage. However, such amount of data can create a real value only if combined with quality: good decisions and actions are the results of correct, reliable and complete data. In such a scenario, methods and techniques for the Data Quality assessment can support the identification of suitable data to process. If for traditional database numerous assessment methods are proposed, in the Big Data scenario new algorithms have to be designed in order to deal with novel requirements related to variety, volume and velocity issues. In particular, in this paper we highlight that dealing with heterogeneous sources requires an adaptive approach able to trigger the suitable quality assessment methods on the basis of the data type and context in which data have to be used. Furthermore, we show that in some situations it is not possible to evaluate the quality of the entire dataset due to performance and time constraints. For this reason, we suggest to focus the Data Quality assessment only on a portion of the dataset and to take into account the consequent loss of accuracy by introducing a confidence factor as a measure of the reliability of the quality assessment procedure. We propose a methodology to build a Data Quality adapter module, which selects the best configuration for the Data Quality assessment based on the user main requirements: time minimization, confidence maximization, and budget minimization. Experiments are performed by considering real data gathered from a smart city case study.

Keywords: `elsarticle.cls`, LaTeX, Elsevier, template
2010 MSC: 00-01, 99-00

*Corresponding author
Email addresses: `danilo.ardagna@polimi.it` (Danilo Ardagna), `cinzia.cappiello@polimi.it` (Cinzia Cappiello), `walter.sama@mail.polimi.it` (Walter Samá), `monica.vitali@polimi.it` (Monica Vitali)

1. Introduction

In the big data era the amount of data is constantly increasing. This is mainly due to datafication that refers to our ability to turn many aspects of our life into digital data and to make a better use of them [1]. In fact, the big challenge of Big Data is to transform the relevant data in good decisions and thus to obtain the maximum value. The problem is that not all the data are relevant: "one of the fundamental difficulties is that extracted information can be biased, noisy, outdated, incorrect, misleading and thus unreliable" [2].

Moreover, Big Data have relevant peculiarities if compared to classic data sources and are usually characterized by errors and missing values. The reliability of the analytics results depends on the reliability of the analyzed information for the specific task. This reliability can be evaluated using Data Quality metrics, which can give a hint to data consumers on how precise their results are expected to be. Since in a Big Data scenario, the implementation of data cleaning approaches is not feasible due to the size and the streaming nature of the data source, the analysis on Data Quality is also useful to select between all the available information, the portion which is more reliable for the analysis, discarding data characterized by poor quality.

Assessing Data Quality (DQ) can be a good starting point for identifying insignificant information. Data Quality is often defined as "fitness for use", i.e., the ability of a data collection to meet users' requirements [3]. It is evaluated by means of different dimensions, which definition mainly depends also on the context of use [4]. Anyway, it is possible to distinguish a small set of DQ dimensions that are considered relevant in most of the studies. Such set includes: accuracy, completeness, timeliness and consistency [4]. For structured data, Data Quality literature offers several contributions proposing assessment algorithms for these consolidated dimensions, but Big Data pose new challenges related to their main characteristics: volume, velocity, and variety. A preliminary discussion on challenges and possible solutions for Data Quality assessment in a Big Data scenario has been proposed in [5]. In particular, in order to address volume and velocity issues, it is necessary to redesign assessment methods for exploiting parallel computing and for reducing the computation space.

In fact, the full assessment of application-dependent Data Quality metrics might require an execution time which is not feasible with the needs of the data consumer in an interactive scenario. As a solution, the Data Quality analysis might be addressed only to a portion of the data, selecting samples of the data source in order to perform the analysis on a reduced amount of time and with a reduced cost.

For this reason, in this paper, we propose to reduce the computation space by considering only a portion of the data as input in order to decrease the time and resources needed. For making the users aware of the reliability of the Data Quality values, we propose to introduce a *confidence* metric as a measure of the data trustworthiness. Confidence is proportional to the amount of considered data that depends on the constraints on the execution time, which in turn is influenced by the amount of available computational resources. Considering

these relations, in this paper we present a Data Quality service able to: (i) evaluate the quality level of big data sets by exploiting parallel computing, (ii) select the amount of data to analyze on the basis of time and resources constraints. We mainly discuss this second feature by presenting the model that explains the relation among confidence, time and cost and showing how it can be used as the basis for an adaptive assessment module that, considering users requirements, is able to automatically change the way in which the evaluation is performed. The paper is structured as follows. Sect. 2 presents a real case study motivating the proposed approach in a smart city public transportation scenario. Sect. 3 discusses the issue of computing Data Quality for Big Data and presents an architecture to support this task. Sect. 4 discusses the proposed approach by defining the parameters and the application scenarios of the proposed Confidence/Cost/Time (CCT) model. Sect. 5 defines the concept of confidence and issues related to its evaluation. In Sect. 6 we present the methodology for building the CCT model, which is applied in the smart city scenario in Sect. 7 and evaluated in Sect. 8. Finally, we discuss related work in Sect. 9 and draw conclusions in Sect. 10.

2. Motivating Example: a Smart City Scenario

The approach proposed in this paper has been conceived by analyzing the scenario and related issues addressed by a joint European and Brazilian project called EUBra-BIGSEA[1]. In this section we describe the goals of this project and the details of the considered data sources in order to better clarify the motivations behind the respective design choices.

The EUBra-BIGSEA project aims to develop a cloud platform fostering Big Data applications development and management. The goal is to design cloud services able to empower Big Data analytics and thus able to support the development of data processing applications. Such services are developed by considering Big Data issues (i.e., data volume, variety, velocity, and veracity), QoS, privacy and security constraints. In particular, the platform is able to manage and store different types of data, such as structured vs unstructured data, stationary vs dynamic data, textual vs numeric data, and to offer a set of Big Data services that can be classified as [6]:

- Data ingestion and stream processing modules: they are in charge to load and synchronize the data stored in the ecosystem.
- Data access and query modules: such modules perform search and filter operations as well as basic aggregations.
- Data analytics and mining modules: they are responsible to run data mining and analytics tasks on the stored data. In particular, services related

[1] http://www.eubra-bigsea.eu/

to predictive and descriptive methods and to Data Quality evaluations are offered.

Such services are provided in order to support users and/or applications in retrieving data stored in the platform and launching value-added analysis.

In order to test and validate all the designed methods, a real user scenario related to smart cities is considered [7]. The project focuses on the analysis of the transportation system of a Brazilian city in order to offer added value services to its citizens. The considered Big Data sources provide stationary geo-referenced data about the city as well as data about dynamic facets of the city life such as GPS location of buses and user cards, weather conditions and social data (i.e., data extracted from social networks). The data will be the input to the Big Data services listed above to provide: (i) support to the citizens by suggesting personalized alternative routes when they are experiencing some travel difficulties; (ii) support the municipalities to have a clear view of the state of the mobility in the city. These data sources are used as input for the data mining algorithms.

In such scenario, the analytics applications need to rely on the data they are analyzing in order to achieve reliable results. The *Data Quality service* presented in this paper is in charge to provide metadata which describe the quality of the sources to support the data mining applications. Through the quality values provided by the proposed approach, data mining applications can have two different benefits:

- awareness of the quality of the result as a consequence of the quality of the input data;
- selection of a proper set of data, which satisfy Data Quality constraints.

The Data Quality assessment is dynamic since it changes whenever the data source is updated. Moreover, the Data Quality evaluation is not general but it is dependent on the use that the data mining application should do of these data and also on the granularity level used by the application. For this reason, the Data Quality assessment should be performed several times for the context of each specific application. In a Big Data scenario this assessment can be computational expensive and a trade off between its accuracy and the computation time and cost is required. In this paper, we discuss this issue by creating a model of the relations existing between the Data Quality assessment confidence (measuring the reliability of the Data Quality computation), the assessment execution time, and the cost of the assessment in terms of required computational resources.

3. Data Quality Assessment in Big Data

As stated in Sect.1, Data Quality research area aims to develop methods for defining, assessing and improving the suitability and usefulness of data for the context in which they are supposed to be used. Many contributions focused on

methods and techniques for dealing with structured data, but Big Data pose new challenges that new methods have to address. This section discusses the main Data Quality concepts and then presents the Data Quality service that we designed within the EUBra-BIGSEA project.

3.1. Data Quality in Big Data

Data Quality is a prerequisite to get valuable results from analytics application. Data Quality is a multidimensional concept since different aspects have to be considered. Such aspects are modeled through Data Quality dimensions that are assessed through determined metrics. The literature presents many Data Quality dimensions but the most used ones are accuracy, completeness, consistency, and timeliness. Accuracy and completeness assess data along their correctness and numerical extension. More formally, *Accuracy* is defined as the proximity of a value v to a value v' considered as correct [8]. *Completeness* is defined as the degree to which a given data collection includes all the expected data values [3]. *Timeliness* evaluates the temporal validity of data and it expresses how "current" analyzed data are. *Consistency* refers to the violation of semantic rules defined over a set of data items.

In Big Data, other dimensions have to be considered. For example, the large number of sources makes trust and credibility important. The *trustworthiness* of a data item is the probability that its value is correct [9] [10] and depends on data provenance. In sensor networks, and thus in a scenario in which sources generate data streams, accuracy has to be considered within *precision* that is the degree to which repeated measurements show the same or similar results [11]. In fact, in case of inaccurate values, precision allows us to detect unexpected context changes and/or malfunctioning of sensors [12]. Also completeness has to be considered from two different perspectives on the basis of the granularity level: the completeness of a specific reading (i.e., a set of values sensed by a sensor at the same time) and the completeness of the data streams. The former assesses the quantity of values received while the latter also considers the situations in which readings can be entirely missing. Such examples show that Data Quality definition and assessment in Big Data mainly depend on data types, data sources, and applications. This is also confirmed in other literature contributions (e.g., [9]). In this paper, we aim to design an adaptive Data quality service that, on the basis of the data input and the applications that request quality value, is able to select the right dimensions and assessment configuration and algorithms.

3.2. Data Quality service

Considering the issues described in the previous section, we propose a solution that has been designed and developed within the EUBra-BIGSEA project described in Sect. 2. Such solution is called *Data Quality service* and mainly provides assessment functionalities.

In this section, we describe the components of the Data Quality service and how the assessment module works also considering Big Data constraints. The proposed *Data Quality Service Architecture* is depicted in Fig. 1.

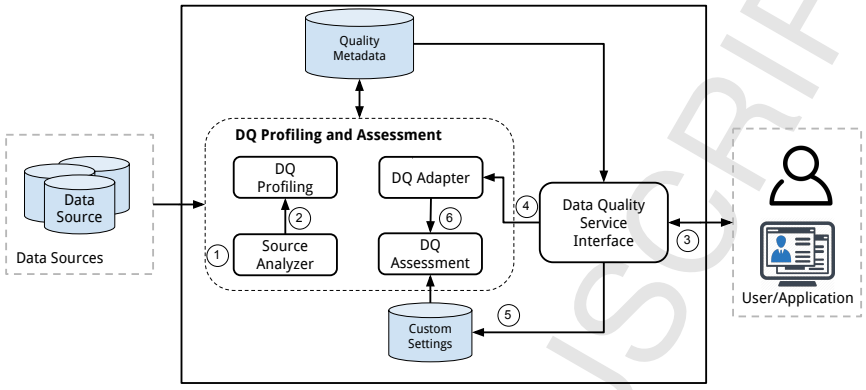


Figure 1: Data Quality Service Architecture

The core of the architecture is the *DQ Profiling and Assessment* module. This module is in charge of collecting statistics and information about data (e.g., types of values, number of repeated values, uniqueness) and it is composed of two main components: the *DQ Profiling* module and the *DQ Assessment* module. The *DQ Profiling* module provides some metrics useful to "measure and monitor the general quality of a dataset" [13]. Profiling is in charge of calculating a set of metadata, which describes the data source and the main features of the fields composing it (e.g., maximum, minimum, average values) and the number of distinct values. This information provides an overview of the data source and can be interesting for each application that intend to use it. The *DQ Assessment* module is in charge of computing Data Quality dimensions. The metrics are computed according to the portion of data selected for the specific application and the quality metrics of interest for the final user. Not all the quality metrics are computable given a data source or a portion of it, since it depends on the nature of the data source and the type of its attributes. In our approach this set is automatically defined when the source is registered to the platform *(1)*. The *Source Analyzer* module automatically detects the kind and structure of the source, the type of its attributes and the quality dimensions that is possible to evaluate, solving the context-dependent Data Quality assessment issue. After that, an initial profiling of the source is executed *(2)*. Note that, especially in case of dynamics sources (e.g., data streams), profiling needs to be performed periodically in order to update over time the status of the considered source. The update rate is defined by the Source Analyzer on the basis of the variability of the source.

The results of all the analysis performed by these two modules are stored in the *Quality Metadata* repository and are mainly used to support the DQ assessment.

DQ assessment is performed on demand. The *Data Quality Service Interface(3)* lets the users and/or applications access the Data Quality service in

order to gather metadata that describe the quality level of the analyzed data sources. Through this interface, (i) users/applications access the characteristics of the data sources (i.e., profiling metadata) and (ii) having an overview of the content, they are able to select and filter the data sources to analyze, the DQ dimensions to evaluate and the granularity with which the quality has to be assessed (i.e., global, attribute or value level). The system collects all the users/applications settings in order to build a *Configuration file* that is used to invoke the Data Quality service and to execute all the requested evaluations *(4)*. Preferences are saved for the subsequent invocations in the *Custom Settings repository (5)*.

When the user request is submitted, it is analyzed by the *DQ Adapter* module, that tunes the precision of the results according to the specification of the user. The adapter addresses the velocity issues if fast responses are needed. In fact, since the Data Quality computation can be time expensive, the adapter can select a subset of the available data to provide a faster evaluation but with a lower precision, that we defined as an index called *Confidence*. Once the confidence level has been established, the DQ Assessment is executed *(6)*.

To conclude, the output of the *DQ profiling and assessment* module is a set of metadata expressing a Data Quality evaluation of the sources, coupled with a precision value. This information is written in the *Quality Metadata* database.

3.3. Data Quality Profiling and Assessment module

The Data Quality Profiling module, in our implementation, provides the following information for all the attributes contained in the Data Object: number of values, number of null values, number of distinct values, maximum, minimum, mean and standard deviation (only for numerical values). The Data Quality assessment module is instead able to evaluate the following DQ dimensions:

- *Accuracy*: it is defined as the degree with which a value is correct [3]. Currently, we have implemented it only for numerical values, in order to check if they are included in an expected interval or they are outliers (and thus not accurate).

- *Completeness*: it measures the degree with which a dataset is complete [3]. It is evaluated by assessing the ratio between the amount of values currently available in the dataset and the expected amount of values. The expected amount of values considers both null values in available registrations and missing registrations. Note that, as regards data streams, missing registrations are easy to detect if data are sensed with a specific frequency. If data are not collected at a regular pace, it is possible to rely on historical data to estimate the sampling frequency that often varies over time.

- *Consistency*: it refers to the violation of semantic rules defined over a set of data items [4]. Therefore, this dimension can be calculated only if there is the availability of a set of rules that represent dependencies

between attributes. We have developed a module that detects functional dependencies and checks if the values in the dataset respect them.

- *Distinctness*: it is related to the absence of duplicates and measures the percentage of unique registrations or distinct attribute values in a dataset.
- *Precision*: this dimension can be calculated only for numerical attributes and can be defined as the degree with which the values of an attribute are close to each other. In particular, precision is derived by considering the mean, and the standard deviation of all the values of the considered attribute.
- *Timeliness*: it is the degree with which values are temporally valid. It is evaluated as described in [14] by considering both the timestamp related to the last update (i.e., currency) and the average validity of the data (i.e., volatility).
- *Volume*: this quality dimension provides the percentage of values contained in the analyzed Data Object with respect to the source from which the Data Object is extracted.

The Data Quality assessment module is able to compute all these metrics for each user/application request. Note that different applications can have different requirements: not all the dimensions are always relevant. For example, let us consider a researcher that aims to analyze available data sources by applying data mining algorithms. The invocation of the DQ service allows him/her to understand the characteristics of the data source, to evaluate the suitability of the quality level of the source to the analysis that s/he aims to perform and to discard the data that do not satisfy her/his requirements. In the specific case, (i) data volume should be significant in order to have significant results, (ii) accuracy and completeness should be high in order to have a correct output, (iii) timeliness is less relevant since usually data mining algorithms are historical evaluations and no specific time constraints are needed. The DQ service is able to evaluate these dimensions and inform the researcher about the appropriateness of the considered sources. The importance of the quality dimensions, and thus the user requirements, also depend on the application context: for example the health care domain requires very high data accuracy and precision since errors can often have disastrous effects. There can be also the situation in which data have to be used by an application that has to provide an actual information to the users on the basis of a current context. Here, a high DQ level is an important requirement together with a fast execution time. In all these situations, the *DQ Adapter* module is in charge of tuning the assessment by defining the confidence level and the number of computational nodes to employ according to the user requirements. In the rest of the paper we focus on the design of the adapter module.

4. Data Quality Adapter Implementation

In this work, we aim to provide an approach for supporting data consumers in the selection of the best settings for evaluating Data Quality in a Big Data environment, taking into account the trade off between execution time, Data Quality evaluation confidence, and cost of the evaluation. This is the goal of the *DQ Adapter* module presented in the Data Quality Service Architecture (Sect. 3).

In this scenario three main parameters need to be considered:

- *Confidence (C)*: the Data Quality evaluation is totally reliable only if the whole dataset is evaluated. The analysis of a subset of the dataset gives some hints on Data Quality but with a reduced confidence. Confidence is proportional to the fraction of the dataset considered in the evaluation (Sect. 5);
- *Execution Time (T)*: this is the time required for the evaluation of the Data Quality of a dataset. This time is expected to increases with the size of the dataset and to decrease with the number of computational nodes involved in the evaluation;
- *Budget (B)*: in an Infrastructure as a Service (IaaS) scenario, the cost for executing the Data Quality computation depends on the execution time and the amount of resources (computational nodes) employed.

These three parameters represent non-functional requirements for the final users and are in contrast with each other. In order to provide a solution taking into account these requirements, it is necessary to build a model of the expected behavior of the Data Quality analysis considering these variables.

The model captures which are the relations between the three parameters involved and can be used to perform an optimization of the non-functional requirements expressed by the user. We identified three main scenarios, according to the main goal of the user:

- **Scenario 1 - Confidence maximization:** in this scenario the user expresses constraints on the maximum execution time and budget, and searches for the configuration with the maximum confidence. The main goal is to obtain the higher confidence given the time and budget constraints, in order to have a reliable evaluation of the Data Quality.
- **Scenario 2 - Time minimization:** in this interactive scenario the user expresses constraints on the maximum budget and the minimum confidence level and explores the model to select the best trade off in order to minimize the execution time. The main goal is to have a fast response on Data Quality evaluation, enabling a responsive interaction between the user and the Data Quality evaluation algorithms.

- **Scenario 3 - Budget minimization:** in this scenario the main concern of the user is the cost of the solution. Given maximum time and minimum confidence constraints, the model is navigated to select the best configuration to obtain budget minimization.

The underlying model, given the constraints expressed in terms of the three non functional variables, supports the user in selecting the best configuration (i.e., confidence level and number of cores to perform the analysis) to run the Data Quality assessment. As will be discussed in Sect. 8, each scenario results in solving an optimization problem given the user requirements, for which several configurations might reach the maximization of the goal of the user while satisfying the constraints.

5. Confidence Evaluation for Data Quality Metrics

To develop the DQ Adapter, first we need to introduce the concept of confidence, which starting from [15] is defined as follows:

Definition 5.1. *The **confidence** is a dimension of "believability", and it can be defined as the extent to which the Data Quality analysis can be trusted with respect to the data at hand. Confidence can be considered a metadimension since it expresses the quality of the Data Quality evaluation and it is linked with the volume and the reliability of the data source.*

The confidence assesses the credibility and correctness of the analysis with respect to the portion of data used such that users and applications can save resources if needed. It depends on the fraction of the dataset analyzed with respect to all the data of the dataset relevant for the application.

In order to reduce the size of the dataset, a sampling algorithm is applied. Choosing the proper sampling technique becomes a critical point in order to be able to derive the best approximation of the reduced dataset compared to the complete one. There are different ways to derive a sample from a dataset [16] [17]:

- *Simple Random Sampling*: is the basic technique with which a sample is selected from a larger group (a population). With Simple Random Sampling, tuples are selected by chance and each one has an equal probability of being included in the sample. In this way, some values of some attributes could never be taken into the samples, so the final quality is inevitably biased. Nevertheless, simple random sampling is the fastest method.

- *Weighted Random Sampling*: unlike simple random, in this method an individual has a known, but possibly non-equal, probability of being included in the sample. For example, using a systematic random sampling, individuals (i.e., tuples in our case) are selected at a regular interval. The probability to be included in the sample is dependent on such interval that may be for example in terms of time or order. That is, in case of data

stream, it is possible to select only the registrations appearing every 25 minutes or we can select every 10th tuple present in a table). In other cases, the probability can be proportional with the usage frequency of a tuple. By using weighted random sampling, the likelihood of bias can be reduced.

- *Stratified Random Sampling*: in order to use this method, it is necessary to identify some factors through which it is possible to divide the population into different groups. Each group should be associated with a measurement of interest that varies among the different sub-groups. This method is usually applied when a population is very heterogeneous.

To minimize the overhead, in this work we adopted the simple random sampling method.

The confidence of the subset extracted through sampling the original dataset is generally expressed as:

$$C = \frac{sampled\_records}{total\_records} \tag{1}$$

even though different Data Quality metrics might be affected in a different way by the confidence. Experiments on this issue are discussed in Sect. 8.1.

6. Non-Functional Requirements Optimization Model

The described context highlights the importance for a user to be aware of the quality of the data she/he is using, but also the complexity and cost of assessing this information with the growth of the dataset size. In this paper we propose a methodology for building a model with the aim of supporting the user in selecting the best configuration of the assessment architecture. An optimization problem arises due to the different possible scenarios that the quality module has to face. The model is based on the provided non-functional requirements:

- T: the Execution Time in minutes,
- B: the Budget in $,
- C: the Confidence level from 0^+ to 1.

Given the specific scenario of interest, the above non-functional requirements will have the role of objective function or decision variables. In this latter case, the non-functional requirements will be subject to a constraint. In the following we will denote with $\bar{\cdot}$ the constraint parameter (e.g., $T \leq \bar{T}$ will introduce a performance constraint predicating on the execution time). The process to build the model consists of three steps:

1. samples collection;
2. confidence model regression;
3. cost-oriented model exploration.

These three steps are discussed in the rest of the section.

6.1. Samples collection

The first step consists in testing the available infrastructure in order to collect data about the relation between the three considered parameters: execution time, confidence level, and budget. In the considered IaaS scenario, the budget is a value which is directly dependent on the number of cores involved in the Data Quality service execution and on the time of execution. According to this, in this initial steps the number of cores is considered in place of the budget.

Data are collected by running the Data Quality service using Spark in a distributed way on several nodes by changing the configuration of the execution. We based our system on Spark because it is one of the most promising framework [18], and thanks to its in memory primitives for Big Data processing, allowed us to obtain an efficient implementation.

More specifically, the algorithm is executed considering:

- different confidence levels: the dataset input of the Data Quality assessment is reduced through sampling in order to evaluate the results with various levels of confidence;
- different number of cores: several configurations are tested by changing the number of cores used to run to the assessment algorithm.

For each configuration, the execution time is measured. The collected samples give hints on how the confidence level and the number of cores affect the execution time of the Data Quality assessment algorithm. Since the described analysis can not be extensive due to the large number of configurations, these data are used as input to build a model representing the dependencies between these three parameters.

6.2. Confidence model regression

From the samples extracted by the previous step, it is possible to build a model, which represents the dependencies among confidence level, execution time, and number of cores. In this paper, we propose to build the model by using a regression method, in order to find the continuous function that better fits the samples collected in the previous phase. To build the model we used the approach described in [19], which shows how it is possible to regress the execution time of a general Hadoop or Spark application with different techniques and it demonstrated that the *linear Support Vector Regression* (SVR) is the most successful one.

SVR is a popular machine learning approach [20], famous for its robustness and insensitivity to outliers. Given the shapes of the curves obtained empirically in the previous step, for a given value of confidence level c, we decided to represent the execution time T_c as a monotonically decreasing functions of the number of cores n:

$$\boldsymbol{T_c(n)} = \frac{\alpha_c}{n} + \beta_c \tag{2}$$

The monotonically decreasing function is a parametrical function with parameters α_c and β_c. Equation (2) is the result of a machine learning process to get a first order approximation of the execution time of Hadoop or Spark jobs in cloud clusters. In order to select a relevant feature set, in [19] we started by generalizing the analytical bounds for MapReduce clusters proposed in [21, 22]. This approach yielded a diverse collection of features including the number of tasks in Spark stage, average and maximum values of task execution times, average and maximum shuffling times, as well as the number of available cores, of which we consider the reciprocal. Since most of these features characterize the Spark application execution, but cannot be controlled, equation (2) collapses all but number of cores, with the corresponding coefficients, into a single constant term, β_c, that is the linear combination of the feature values with the SVR-derived weights.

As it will be discussed in Sect. 7.2, the accuracy has been evaluated through the Mean Absolute Percentage Error (MAPE). The average error obtained is equal to 7%, which is usually considered reasonable for cluster management and capacity planning [23].

6.3. Cost-oriented model exploration

The result of the previous step provides means to model the relations among the time, confidence, and cores dimensions. One of the non-functional requirements of the proposed approach is the budget required to support the assessment execution. In order to include the budget in our model, we need to map the non-functional requirements dimensions considered to the cost of the solution. In this step we transform the model in a Confidence/Cost/Time model, referred as CCT model, as discussed in Sect. 4.

The cost is a function of the number of computational nodes involved in the process and the execution time. According to the infrastructure provider and to the service contract it is possible to have two different pricing policies:

- *discrete hourly cost* where the user is charged for using the physical infrastructure on a hourly basis;
- *time continuous cost* where the user is charged for the actual time during which the analysis have been executed.

The employed cost model affects in a significant way the choice of the best configuration.

For the *discrete hourly cost* policy, knowing the hourly price of each cluster of machine we can derive the cost function as:

$$\boldsymbol{C(n)} = \lceil \frac{n}{VMConfigCores} \rceil \times cost^h \times \lceil \frac{T}{3.6 \times 10^6} \rceil \tag{3}$$

where n is the considered number of cores, $cost^h$ is the hourly price of the Virtual Machine (VM) used, $VMConfigCores$ is the number of cores available in such VM, and T is obtained by Equation 2 (and measured in milliseconds).

A ceiling operator is applied to the first term, which represents the number of VMs involved in the execution. This is motivated by the fact that, once we use even a single core, we need to pay for the whole VM. According to this, the cost depends on the number of required VMs, rather than on the cores that will be effectively used. Another ceiling operator is applied to the last term for modeling the case in which the partial hour rent is charged as a full hour. It can be removed otherwise.

For the *time continuous cost* policy, the actual cost per second is available. In this case, the cost function can be expressed as:

$$\boldsymbol{C(n)} = \lceil \frac{n}{VMConfigCores} \rceil \times cost^s \times T \tag{4}$$

where $cost^s$ is the IaaS cost for a second, and it can be obtained from the hourly cost as:

$$cost^s = \frac{cost^h}{3.6 \times 10^3} \tag{5}$$

The cost models defined in this section are used to transform the previous Confidence/Cores/Time model in the desired Confidence/Cost/Time model, according to the pricing policy.

7. Building the CCT Model in a Smart City Scenario

In this section, we describe the experimental setup used to test our approach by performing real analysis on a real use case.

In order to build the model discussed in Sect. 6, we collected the samples by executing the Data Quality assessment algorithms on a Big Data source coming from the real world where sensible data were anonymized. The analysis has been executed on the *BusUsers* dataset, collecting the information about ticket validations of the users in the city of Curitiba. The validation is executed using a magnetic card associated with a code. Each line in the log contains: (i) CODELINHA: the code assigned to the bus line in which the ticket validation has been performed; (ii) NOMELINHA: the name assigned to the bus line; (iii) CODEVEICULO: the code assigned to the monitored vehicle; (iv) NUMCARTAO: the code associated with the user magnetic card; (v) DATAUTILIZACAO: the timestamp at which the ticket validation has been recorded. We take into considerations several of the Data Quality metrics (Completeness, Consistency, Distinctness, Timeliness, and Volume) introduced in Sect. 3.3 at different granularity levels (the whole dataset, a single attribute and a single value for an attribute of a data source).

To perform these tests a Microsoft Azure Cluster with Xeon processors, from 8 cores up to 48 cores @3GHz and from 12Gb up to 52Gb of RAM, has been used based on D4 v2 VMs. The cluster was run with 1 up to 6 workers with 4 executors per worker, each one with 2 cores and 2Gb of RAM, and a driver, the master node of the Spark application, with 4Gb of RAM.

For what concerns the evaluation of the accuracy of the estimates of the execution time, and also of the comparison between the actual quality dimensions and the one obtained at the different confidence levels (due to sampling) we consider the Mean Absolute Percentage Error (MAPE). For evaluating the confidence precision, the MAPE is defined as:

$$\boldsymbol{MAPE} = \left(\frac{1}{\mathbf{card}(C)} \sum_{c \in C} \frac{|Actual_c - Forecast_c|}{|Actual_c|} \right) \tag{6}$$

where C is the set containing all the available confidence values, $\mathbf{card}(C)$ represents its cardinality, $Actual_c$ and $Forecast_c$ represent the real value of a given quality metric (e.g., Completeness, Distinctness) or the time measured on the system and the predicted one for the considered confidence level (or via Eq. 2).

7.1. Step 1: samples collection

The first step of the approach consists in collecting sample data from the execution of the *DQ Assessment* module with different configurations. The algorithm has been executed several times on the described environment with different configurations in order to collect samples to build the non-functional requirements dependency model. In our tests we considered 64 configurations, obtained by combining these variables:

- Number of cores: 12, 16, 20, 24, 32, 38, 40, 48.
- Confidence levels: from 0.125 to 1.000 with step 0.125.

Starting from 1 up to 6 workers, with step 1, the full analysis is repeated 2 times, the first time all the available cores are used and the second time half of the available cores are used. Moreover, for each configuration, eight levels of Confidence are tested, starting from 12.5% to 100% with pace 12.5%. For each configuration, the analysis has been repeated three times and by aggregating the execution times of the three repetitions, a triple (Time, Cores, Confidence) has been obtained.

Fig. 2 shows a graphical representation of the triples by connecting the points with the same confidence level. As can be observed, the results follow a monotonically decreasing function with a minimum execution time of 35 minutes when considering the lowest confidence with the highest number of cores and with a maximum of 2 hours and 20 minutes in the opposite case.

The tests executions have been affected by some noise for the configurations with 24 and 38 nodes, resulting in a very high execution time. Noisy data, which differ from the average behavior for more than three times the standard deviation, have been removed from the samples set. This issue is independent from our approach but it is seldom observed in a cloud environment as analyzed in [24].

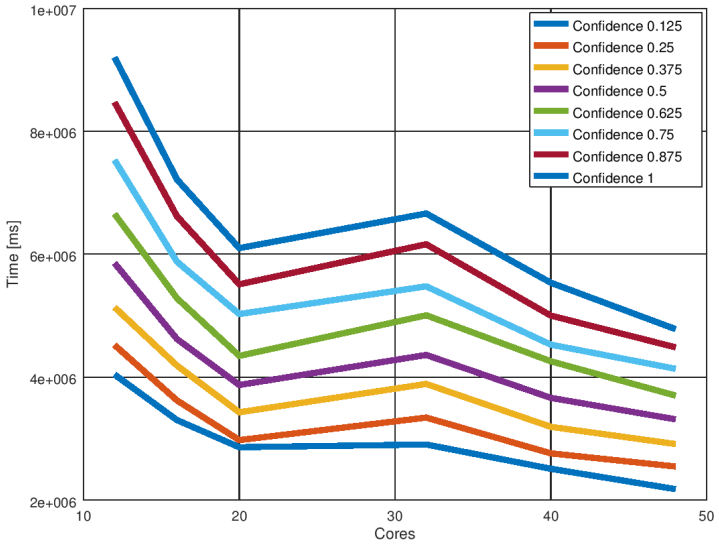


Figure 2: Empirical execution time vs. number of cores dependency at multiple confidence levels.

7.2. Step 2: confidence model regression

In this step, we have applied the confidence model regression described in Sect. 6.2 to the samples obtained in the previous step. For each confidence level, the parameters α_c and β_c have been obtained by considering SVR and by splitting the dataset gathered during the profiling into training and test set (80-20 ratio has been considered). The values obtained for each confidence level are listed in Tab. 1. They are expressed by large numbers because the considered execution times were in measured in milliseconds.

In Fig. 3 the results of the regression model are plotted in comparison to the curves obtained from the interpolation of the sampling results and reported also in Fig. 2. Due to the different slope of the initial segment for the different confidence values, in the derived curves of Fig. 2, with respect to the other ones, we decided to put some higher weights (five times larger) in the training model for the configurations with 12 and 16 cores. In this way, we force the training function to focus more on few cores to derive the final model.

The model has been evaluated by computing the MAPE error for each confidence level on the test set. The resulting average MAPE is equal to 7%, demonstrating a good accuracy for the derived model.

7.3. Step 3: cost-oriented model exploration

As discussed in Sect. 6.3, the last step consists in transforming the Confidence/Cores/Time model in a Confidence/Cost/Time model by applying Eq. 3

Confidence c	α_c	β_c
0.125	21480000	1964776
0.250	23348496	2168536
0.375	27011760	2507483
0.500	29428656	2917189
0.625	33823152	3401108
0.750	39272400	3484089
0.875	46153008	3832474
1.0	50145744	4181366

Table 1: Parameters of the execution time regression model

or Eq. 4 according to the pricing policy. In the experiments, we used the Azure Microsoft cloud as IaaS with configuration D4 v2. At the time of the experiments, the hourly price was equal to *0.458 dollars/hour*[2]. In the experiments, we applied both pricing policies to show the impact of the policies on the optimal configuration selection.

Given the model and the non-functional requirements, selecting the best value for the three dimensions (confidence, cost, and execution time) becomes an optimization problem. In Sect. 8.2 we discuss the application of the model in the three scenarios introduced in Sect. 4.

8. Experimental Evaluation

In this section we discuss the results of the experiments. First of all we discuss how the confidence level affects the several Data Quality metrics comparing the values for the DQ dimensions obtained considering the whole dataset with the one obtained sampling the dataset according to the different confidence levels. Then, we apply the model to the three scenarios introduced in Sect. 4 and we discuss how the obtained CCT model can support the configuration selection.

8.1. Confidence impact on Data Quality assessment

In this section we will discuss the quality results of each different DQ Assessment test executed, in order to find out how the quality changes based on the considered confidence level. For each Data Quality dimension, the assessed value of the whole dataset is compared with the ones obtained with the sampled datasets for each confidence level. As discussed in Sect. 7.1, each test has been

[2] `https://azure.microsoft.com/en-us/pricing/details/virtual-machines/linux`, under General Purpose Machines. Last Accessed 30-03-2017

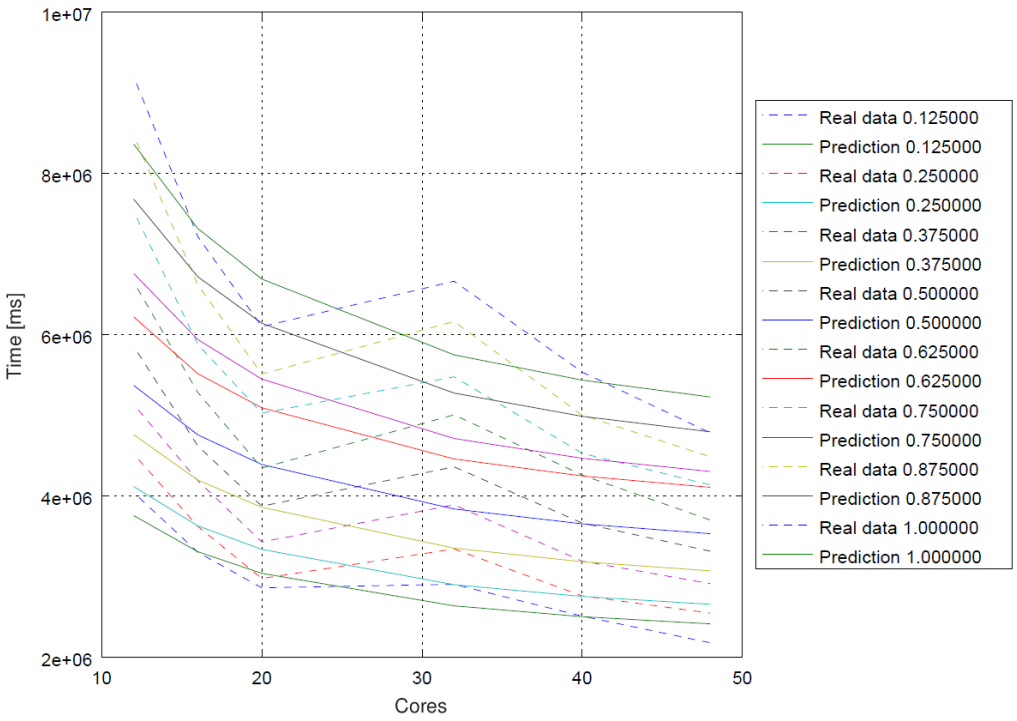


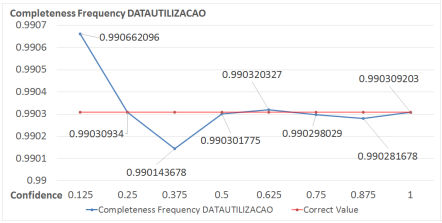
Figure 3: Empirical execution time vs. SVR fitting at multiple confidence levels.

repeated three times and the obtained results are the average of the results of each execution. The maximum error is expressed by considering each confidence level with respect to the real value, and computing the MAPE. This evaluation is important because it supports the user in the selection of the proper confidence level: the higher the sensitivity of the metric to the volume of considered data in the evaluation, the higher the required confidence to obtain reliable data; the lower the sensitivity, the lower the required confidence.

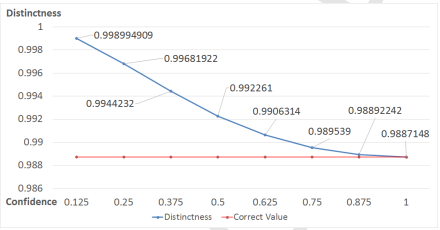
The experiments discussed in this section show that the confidence is not the only dimension to be considered when evaluating the reliability of the analysis. In fact, the confidence has a different impact on different metrics. Also, this impact is dependent on the dataset considered.

Here we make some considerations by using the experiments on the smart city scenario, and specifically analyzing the *BusUsers* data stream.

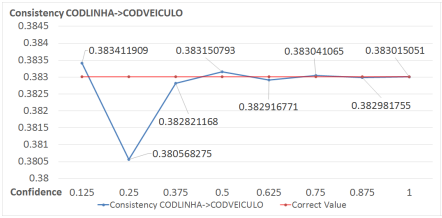
Completeness. The comparison of the evaluation of the completeness metric with the full dataset and with the different confidence levels resulted, on average, in a MAPE of 0.007% (Fig. 4a). The maximum error measured is equal to 0.03% and it can be reduced to 0.01% whenever at least a Confidence equal to 0.5 is chosen. For this metric, the confidence level has a limited effect on the reliability



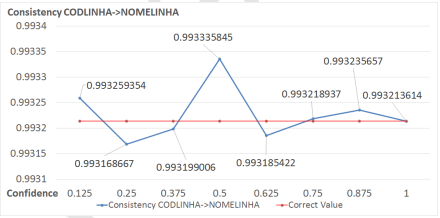
(a) Completeness confidence evaluation



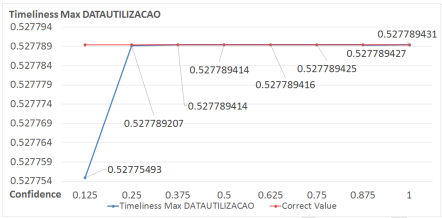
(b) Distinctness confidence evaluation



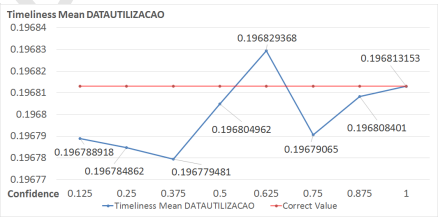
(c) Consistency confidence evaluation (rule 1)



(d) Consistency confidence evaluation (rule 2)



(e) Timeliness max confidence evaluation



(f) Timeliness mean confidence evaluation

Figure 4: Confidence impact on Data Quality assessment dimensions

of the evaluation.

Distinctness. The evaluation of the distinctness metric does not change significantly with respect to the considered confidence. The resulting average MAPE is equal to 0.384% and the maximum error is equal to 1% (Fig. 4b). We obtain a monotonically decreasing distribution of the values and this can be explained by considering that, by taking the samples randomly, it is less probable to get consecutive rows and so it is less probable to obtain duplicates with respect to a sequential random sampling.

Consistency. The consistency of the sampled dataset is compared with the one of the whole dataset for each of the association rules detected. As an example, we evaluated the consistency of the rule $CODLINHA \rightarrow CODVEICULO$, which specifies the existence of a semantic rule between the code of the bus line and the code of the vehicle. If the rule is verified, then a vehicle is mainly assigned

to a single bus line. The results are very similar with each other (Fig. 4c), with a MAPE equal to 0.109% and a maximum error equal to 0.3%.

The consistency reliability is dependent on the association rule. The rule $CODLINHA \rightarrow NOMELINHA$ specifies the existence of a semantic rule between the code of the bus line and the name of the bus line. If the rule is verified, than a bus line name is associated with a single bus line code. In this case, the obtained MAPE is equal to 0.004% and the maximum error is less than 0.01% (Fig. 4d).

Timeliness. The timeliness is evaluated by the *DQ Assessment* module computing three values: the minimum, maximum, and mean timeliness of the dataset. All the minimum values are equal to 0 in this source and so we will not show them in details, even if theoretically a random sampling can take only the most recent values in the worst case. For the maximum value, a small amount of random data are sufficient to derive a very good approximation of the real value: in fact, the MAPE is equal to 0.001% and the maximum error is equal to the 0.003% and the latter one can be also reduced to 0.00002% if a confidence level greater or equal to 0.250 is considered (Fig.4e).

For what concerns the mean Timeliness, its values are strictly dependent on the samples. For this metric, the MAPE is equal to 0.001% and the maximum measured error is equal to the 0.004% and so it can be considered still very low (Fig.4f).

8.2. Applying the CCT model to the different user scenarios

In this section, the CCT model has been applied to the three scenarios driven by the user goals: (i) Confidence Maximization; (ii) Time minimization, and (iii) Budget minimization. The same model is used for each scenario, and axis rotation is applied to put emphasis on the variable of interest for the user. For each scenario, both the discrete and continuous pricing policies are discussed.

8.2.1. Scenario 1: Confidence maximization

In this scenario, the confidence is expressed as a function of the Execution Time and the Budget. The optimization problem to solve will have as objective function the confidence maximization, and constraints on the execution time and budget according to the user requirements:

$$\max_{T,B} C = f(T, B)$$

$$\text{subject to:}$$

$$T \leq \bar{T}$$

$$B \leq \bar{B}$$

The CCT model can be used to solve this optimization problem formulation, which after some algebra, allows to obtain a relation among the three variables in closed form. The result will be the configuration, which allows to get the maximum confidence with the time and budget constraints expressed.

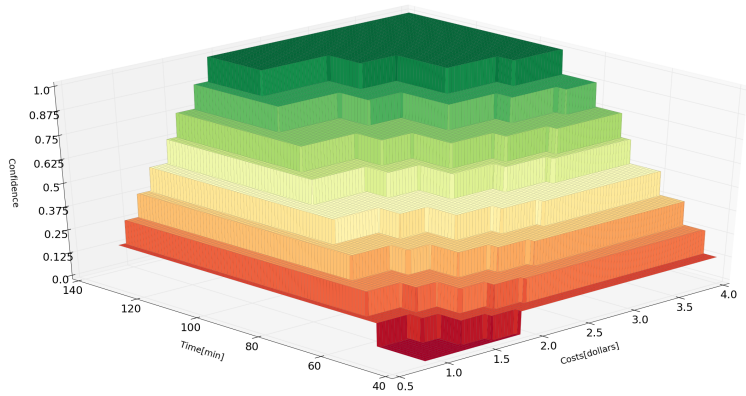


Figure 5: Confidence Maximization with time continuous cost.

In order to respect the constraints, all the configurations with $T > \bar{T}$ and with $B > \bar{B}$ are ignored. Given the model, several solutions might provide the maximum confidence level. These can be considered as *Pareto* points belonging to the *Pareto Frontier*: each point of the Pareto front satisfies the requirements maximizing the confidence. The user can choose the configuration between them according to which of the other parameters has a higher importance to her/him. In our approach, the user can additionally specify which requirement to further optimize (e.g., to minimize the budget or the execution time). Now we analyze the results using the two different pricing policies.

Time continuous cost. In this scenario we applied the hourly cost to the model using Eq. 4. In this case costs and times are evenly distributed and the results are shown in Fig. 5.

The discrete maximum confidence function is represented, as expected, as a 3D step-style graph from which we can make some considerations:

- with $\bar{B} < 1\$$ and $\bar{T} < 50$ *minutes* the full analysis can not be executed since there is no confidence level which satisfy those constraints;
- at least 1$ has to be spent in order to enable the analysis. However, less than 2$ are sufficient to get the results with the maximum confidence in 120/140 minutes;
- at least 35-40 minutes are needed to perform the analysis with a confidence level greater or equal than 0.25, but to get $C = 1$ the module should run more than 80 minutes, even if we are inclined to spend more than 3$;
- there are multiple configurations that enables to reach the maximum confidence. For example in addition to the configurations discussed above we can achieve it with 3$ and 100 minutes, or with 2.7$ and 110 minutes.

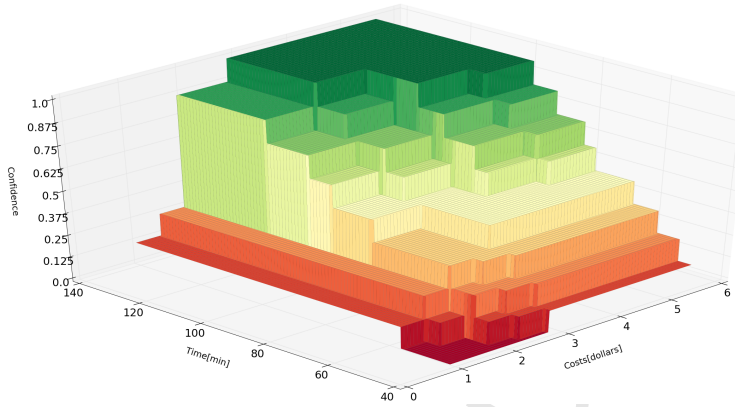


Figure 6: Confidence Maximization with discrete hourly cost.

Discrete hourly cost. In this scenario, the usage of the nodes is charged per hour. The model obtained in in this scenario is shown in Fig. 6.

Observing the graph some considerations can be made:

- when $\bar{T} \leq 1$ *hour* it is impossible to analyze the whole dataset with a confidence greater than 0.25 without having to pay at least 2$;
- the Pareto points for $C = 1$, that were previously obtainable without spending more than 2.5$, are not available in this scenario;
- costs are generally higher than the previous considered case, with a 5.5$ peak with respect to the previous 4$ peak;
- in some cases, by slightly increasing the costs, multiple steps of confidence can be crossed.

Comparisons of the two pricing policies. To highlight the impact of the pricing policies on the optimal configuration we compared the models shown in Fig. 5 and Fig. 6 plotting a graph representing the difference between them, shown in Fig. 7.

The main difference in confidence, equal to 0.625, can be found between the costs equal to 1$ and 2$ and the execution time equal to 120 and 140 minutes. This can be explained by considering that such solution corresponds to the configurations with 12 and 16 cores whose costs is 0.916$/h. In case the analysis lasts 140 minutes, the cost with the discrete hourly cost policy is equal to 2.748$, instead of the 2.173$/h of the time continuous cost policy. From the graph, we can observe that with a budget difference of 0.575$/h, we can obtain a better confidence level than before only by using a different configuration, maintaining the same final budget and the same final execution time.

The other main differences can be observed in the interval of execution time from 1 to 2 hours, as we can expect from the previous reasoning.

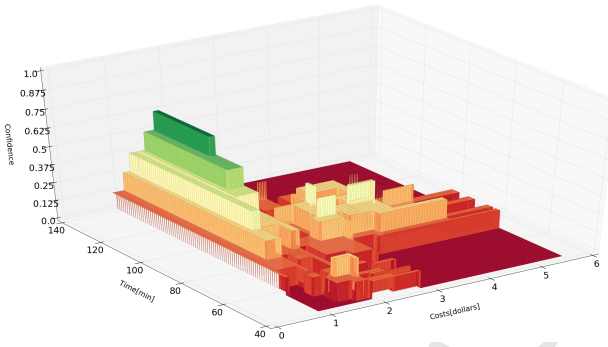


Figure 7: Scenario 1: comparison between the two pricing policies.

The points represented in dark red in Fig. 7 represents configuration in which the difference between the pricing policies in terms of confidence levels is equal to 0.

8.2.2. Scenario 2: Time minimization

In this scenario the graph is rotated to highlight the execution time dimension. Time minimization represents our objective function and requirements on confidence and budget are the model constraints. In solving the problem, all the configurations with $C < \bar{C}$ and $B > \bar{B}$ are ignored. The optimal configuration minimizing the execution time is selected among the ones that satisfy the previous requirements.

Time continuous cost. As shown in Fig. 8, in this scenario the minimum execution time and the minimum cost correspond with a confidence level equal to 0.125. This is the configuration with the lowest execution time, without considering the points in which the analysis cannot be performed, that are represented in dark red. Moreover, the higher is the confidence level, the higher will be the execution time by considering the cost as fixed.

Discrete hourly cost. In this scenario, depicted in Fig. 9, the differences of the solutions with a cost equal to 2$ or 3$ are less evident. It is worth noticing that it is not possible to complete an analysis with $C = 1$ without spending less than 2.7$, which is the configuration requiring the maximum execution time.

Comparisons of the two pricing policies. The graph representing the difference between the two pricing policies for scenario 2 is shown in Fig. 10. The points with a positive difference represent all the configurations possible only with a continuous cost model.

As expected, the discrete hourly cost policy represents only a limitation without benefits. Whenever the cloud system has this second type of pricing

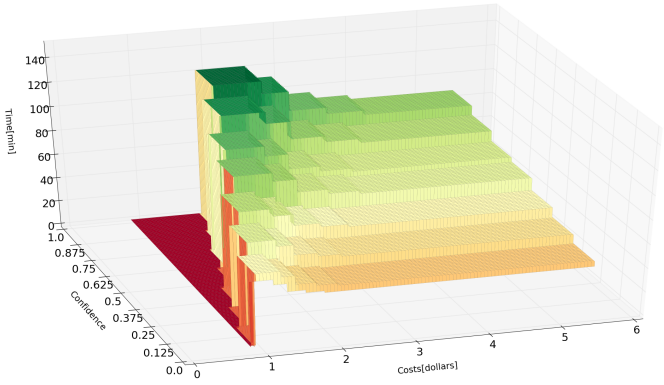


Figure 8: Time minimization with time continuous cost.

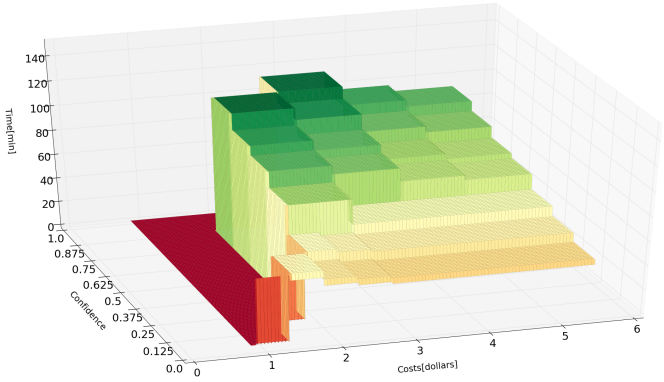


Figure 9: Time minimization with discrete hourly cost.

policy, we can only suggest to users to specify as requirements the points represented in light red, in the previous figure, in which the difference between the policies is near to 0.

8.2.3. Scenario 3: Budget minimization

In this scenario, the objective function of the optimization problem is the minimization of the budget, while confidence and execution time are constraints such that $C \geq \bar{C}$ and $T \leq \bar{T}$. The Pareto frontier is composed of all the points satisfying the constraints with the minimum budget value.

Time continuous cost. The graph that we obtain with this scenario is shown in Fig. 11. The shapes of the cost functions for each confidence level are similar to each other, but increasing the confidence, the execution time increases accordingly.

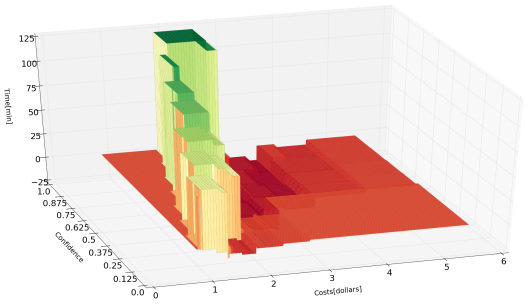


Figure 10: Scenario 2: comparison between the two pricing policies.

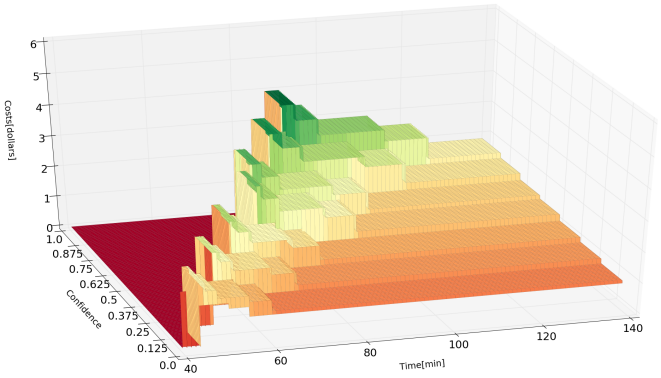


Figure 11: Budget minimization with time continuous cost.

Discrete hourly cost. By considering the discrete hourly cost policy we obtain the graph in Fig. 12.

In this case, by changing execution time and confidence we obtain several solutions with the same budget. In fact, in this scenario, the graph is more flat than in the previous case. For example, this happens for execution time between 60 and 140 and confidence level between 0.375 and 0.75 with a low variations of costs.

Comparisons of the two pricing policies. Even in this case, we depict the difference between the two policies in Fig. 13.

From this graph we can observe that the difference in terms of final costs can be really high, with peaks of 2$ and 2.5$, but there is a dark green rectangle that represent the confidence level equal to 0.875 and the execution time greater than 100 minutes in which the costs of the different types of prices are the same.

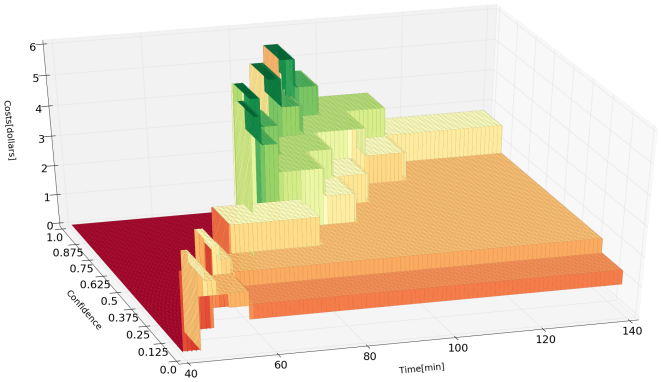


Figure 12: Budget minimization with discrete hourly cost.

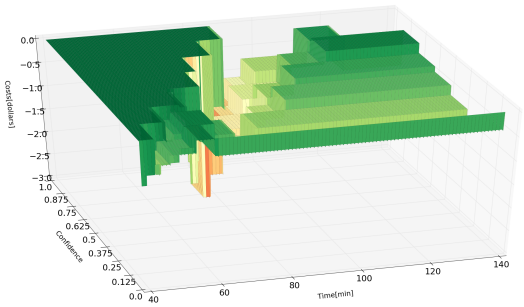


Figure 13: Scenario 3: comparison between the two pricing policies.

9. Related Work

Data Quality research area proposes many consolidated approaches that work fine on relational models but that cannot be properly applied in Big Data environments: Big Data require new models, methods and techniques for the definition, assessment and improvement of Data Quality dimensions. In the literature several papers claim that Data Quality dimensions need to be redefined for Big Data. For example, [9] focuses on the evolution of Data Quality dimensions and shows how the definitions of these dimensions change on the basis of data type, sources and applications considered. Data Quality dimensions have been analyzed in the Big Data scenario also in [25]. Authors also define a quality assessment process for Big Data: the definition of the quality dimensions here depends on the goals of data collection and thus also on the considered business environment and the involved data sources. A model that can be used to assess the level of quality- in-use of the data in Big Data is proposed in [26]. Authors define the concept of adequacy of data as "the state or ability of data

of being good enough to fulfill the goals and purposes of the analysis". Other papers focus only on the novel dimensions that should be introduced in the Big Data scenario. Authors in [27] discuss the rise of Big Data on cloud computing and depict Data Quality as a challenge. In particular, authors state that high-quality data in the cloud are characterized by data consistency: the quality of different data sources is high if there are not inconsistencies among their values. The importance of trustworthiness in the Big Data scenario is highlighted in [28]. Trust together with accuracy has been also considered in [29]. In this paper authors focus on data mining systems and claim that in Big Data the data sources are of many different origins, not all well- known, and not all verifiable. Therefore, data validation and provenance tracing become more than a necessary step for analytics applications. All these papers confirm the motivations behind our work: Data Quality dimensions definition and assessment algorithm have to be redefined and are strongly dependent on the type of data and data source and on the application that requests data. In this work, we do not aim to provide novel definition of quality dimensions but we define an architecture for an adaptive Data Quality service able to provide the right quality metadata for the considered application. Moreover, we show the Confidence impact on the sensitivity of other Data Quality dimensions. In this work we also propose to manage the Data Quality assessment by optimizing non functional requirements such as the accuracy of the evaluation, execution time, and cost, considering a Big Data execution environment. Other approaches have focused on estimating the execution time of applications for cluster capacity planning and/or runtime scheduling. In [30], the authors analyze the performance of Spark applications deployed in a public cloud infrastructure to build models able to estimate the time required by the application. The approach consists in a profiling phase where the application is executed several times with different inputs, and in a testing phase in which the acquired model is used to predict the execution time of the application analyzing the whole dataset. The model aims at capturing the relation between the execution time and the dataset size as well as the configuration parameters. The authors investigated the issue also in [31], where a fluid Petri net has been employed to create a model able to envision MapReduce jobs execution time.

Other approaches focus on performance optimization of map reduce jobs. In [32], the performance of Big Data applications are analyzed in order to detect the main factors that affect their quality, with the aim of detecting the source of the degradation of the applications as well as the limitations of the infrastructure hosting it. In [33], the application non functional requirements are considered by proposing a new cloud service for scaling the infrastructure to meet them. Performance are also studied in [34], where the authors propose a correlation-based performance analysis to identify critical outliers by correlating different phases, tasks, and resources.

10. Conclusions

In this paper, we have analyzed the issue of providing a Data Quality assessment service for supporting analytics application in the selection of a proper set of input while respecting non functional requirements such as execution time and budget constraints.

The goal of the paper is to provide a Data Quality service for applications aiming at analyzing Big Data sources. We proposed an architecture for managing Data Quality assessment and we have focused on the *DQ Adapter* module. This module is designed with context-aware methodology to support the user in selecting the best configuration parameters to run the Data Quality assessment according to the main goal (budget minimization, time minimization, confidence maximization). To support this decision we built a model that we called CCT (Confidence/Cost/Time) model, able to capture the relations between the non functional requirements. The model can be used to solve the configuration selection in terms of number of cores involved in the evaluation and confidence level of the input dataset, as an optimization problem. To model the execution cost, two different pricing models have been considered: the hourly pricing and the time continuous pricing. From the proposed model of Confidence we are able to automatically run the analysis with the best possible parameters based on the non-functional requirements of the users and the applications.

We applied our methodology to a smart city scenario, by analyzing the quality of streaming data collected in the city of Curitiba, and we studied the effect of confidence on the Data Quality assessment. In our case study, the precision of the model to capture the relations between the non functional requirements showed high, with a MAPE lower than 7%. In the experiments, we have demonstrated that the sensitiveness of the Data Quality assessment to the confidence depends both on the data source features and on the specific Data Quality dimension: different dimension can be sensitive in a different way to the confidence level. Moreover, the three optimization scenarios have been analyzed by also comparing the effect of the pricing policy on the configuration selection.

Acknowledgments

The authors work has been partially funded by the EUBra-BIGSEA project by the European Commission under the Cooperation Programme (MCTI/RNP 3rd Coordinated Call), Horizon 2020 grant agreement 690116.

References

[1] X. L. Dong, D. Srivastava, Big Data Integration, Synthesis Lectures on Data Management, Morgan & Claypool Publishers, 2015. doi:10.2200/S00578ED1V01Y201404DTM040.
URL http://dx.doi.org/10.2200/S00578ED1V01Y201404DTM040

[2] L. Berti-Equille, J. Borge-Holthoefer, Veracity of Data: From Truth Discovery Computation Algorithms to Models of Misinformation Dynamics, Synthesis Lectures on Data Management, Morgan & Claypool Publishers, 2015. doi:10.2200/S00676ED1V01Y201509DTM042.
URL http://dx.doi.org/10.2200/S00676ED1V01Y201509DTM042

[3] R. Y. Wang, D. M. Strong, Beyond accuracy: What data quality means to data consumers, J. of Management Information Systems 12 (4) (1996) 5–33.
URL http://www.jmis-web.org/articles/1002

[4] C. Batini, M. Scannapieco, Data and Information Quality - Dimensions, Principles and Techniques, Data-Centric Systems and Applications, Springer, 2016. doi:10.1007/978-3-319-24106-7.
URL http://dx.doi.org/10.1007/978-3-319-24106-7

[5] C. Cappiello, W. Samá, M. Vitali, Quality awareness for a Successful Big Data Exploitation, in: 22nd International Database Engineering & Applications Symposium (IDEAS 2018), ACM, in press, pp. 1–15.

[6] S. Fiore, D. Elia, W. dos Santos Filho, C. E. Pires, D4.1 - design of the integrated big and fast data eco-system, Deliverable, EUBra-BIGSEA Project (H2020-690116) (2016).

[7] T. B. Araújo, C. Cappiello, N. P. Kozievitch, D. G. Mestre, C. E. S. Pires, M. Vitali, Towards reliable data analyses for smart cities, in: Proceedings of the 21st International Database Engineering & Applications Symposium, ACM, 2017, pp. 304–308.

[8] T. C. Redman, Data quality for the information age, Artech House, 1996.

[9] C. Batini, A. Rula, M. Scannapieco, G. Viscusi, From data quality to big data quality, J. Database Manag. 26 (1) (2015) 60–82. doi:10.4018/JDM.2015010103.
URL http://dx.doi.org/10.4018/JDM.2015010103

[10] C. Dai, D. Lin, E. Bertino, M. Kantarcioglu, An approach to evaluate data trustworthiness based on data provenance, in: Secure Data Management, 5th VLDB Workshop, SDM 2008, Auckland, New Zealand, August 24, 2008, Proceedings, 2008, pp. 82–98. doi:10.1007/978-3-540-85259-9_6.

[11] ISO, ISO/IEC Guide 99-12:2007 International Vocabulary of Metrology, Basic and General Concepts and Associated Terms (2007).

[12] C. Cappiello, F. A. Schreiber, Quality and energy-aware data compression by aggregation in WSN data streams, in: Seventh Annual IEEE International Conference on Pervasive Computing and Communications - Workshops (PerCom Workshops 2009), 9-13 March 2009, Galveston, TX, USA, IEEE Computer Society, 2009, pp. 1–6. doi:10.1109/PERCOM.2009.

4912866.
URL http://dx.doi.org/10.1109/PERCOM.2009.4912866

[13] F. Naumann, Data profiling revisited, SIGMOD Rec. 42 (4) (2014) 40–49. doi:10.1145/2590989.2590995.

[14] M. Bovee, R. Srivastava, B. Mak, A conceptual framework and belief-function approach to assessing overall information quality, in: Proceedings of the 6th International Conference on Information Quality, Boston, MA, September 2001.

[15] L. L. Pipino, Y. W. Lee, R. Y. Wang, Data quality assessment, Commun. ACM 45 (4) (2002) 211–218. doi:10.1145/505248.506010.
URL http://doi.acm.org/10.1145/505248.506010

[16] F. Olken, D. Rotem, Random sampling from database files: A survey, in: Z. Michalewicz (Ed.), Statistical and Scientific Database Management, Springer Berlin Heidelberg, Berlin, Heidelberg, 1990, pp. 92–111.

[17] V. Easton, J. McColl, Statistics glossary - version 1.1 (1997).
URL http://www.stats.gla.ac.uk/steps/glossary/sampling.html

[18] H. Derrick, Survey shows huge popularity spike for Apache Spark (2015).
URL http://fortune.com/2015/09/25/apache-spark-survey

[19] E. Ataie, E. Gianniti, D. Ardagna, A. Movaghar, A combined analytical modeling machine learning approach for performance prediction of mapreduce jobs in cloud environment, in: 18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2016, Timisoara, Romania, September 24-27, 2016, 2016, pp. 431–439. doi:10.1109/SYNASC.2016.072.
URL http://dx.doi.org/10.1109/SYNASC.2016.072

[20] V. N. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[21] A. Verma, L. Cherkasova, R. H. Campbell, ARIA: Automatic Resource Inference and Allocation for Mapreduce Environments, in: Proc. ICAC 2011, 2011.

[22] M. Malekimajd, D. Ardagna, M. Ciavotta, A. M. Rizzi, M. Passacantando, Optimal map reduce job capacity allocation in cloud systems, SIGMETRICS Perform. Eval. Rev. 42 (4) (2015) 51–61.

[23] E. Lazowska, J. Zahorjan, G. Graham, K. Sevcik, Quantitative system performance: computer system analysis using queueing network models, Prentice-Hall, Inc., 1984.

[24] D. Ardagna, G. Casale, M. Ciavotta, J. F. Pérez, W. Wang, Quality-of-service in cloud computing: modeling techniques and their applications, Journal of Internet Services and Applications 5 (1) (2014) 11. doi:10.1186/s13174-014-0011-3.
URL http://dx.doi.org/10.1186/s13174-014-0011-3

[25] L. Cai, Y. Zhu, The challenges of data quality and data quality assessment in the big data era, Data Science Journal 14 (2016) 2.

[26] J. Merino, I. Caballero, B. Rivas, M. A. Serrano, M. Piattini, A data quality in use model for big data, Future Generation Comp. Syst. 63 (2016) 123–130. doi:10.1016/j.future.2015.11.024.

[27] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, S. U. Khan, The rise of "big data" on cloud computing: Review and open research issues, Inf. Syst. 47 (2015) 98–115. doi:10.1016/j.is.2014.07.006.
URL http://dx.doi.org/10.1016/j.is.2014.07.006

[28] E. Bertino, Data Trustworthiness—Approaches and Research Challenges, Springer International Publishing, Cham, 2015, pp. 17–25.

[29] D. Che, M. Safran, Z. Peng, From Big Data to Big Data Mining: Challenges, Issues, and Opportunities, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 1–15.

[30] G. P. Gibilisco, M. Li, L. Zhang, D. Ardagna, Stage aware performance modeling of dag based in memory analytic platforms, in: Cloud Computing (CLOUD), 2016 IEEE 9th International Conference on, IEEE, 2016, pp. 188–195.

[31] E. Gianniti, A. Rizzi, E. Barbierato, M. Gribaudo, D. Ardagna, Fluid petri nets for the performance evaluation of mapreduce applications, in: proceedings of the 10th EAI International Conference on Performance Evaluation Methodologies and Tools on 10th EAI International Conference on Performance Evaluation Methodologies and Tools, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2017, pp. 243–250.

[32] L. E. B. Villalpando, A. April, A. Abran, Performance analysis model for big data applications in cloud computing, Journal of Cloud Computing 3 (1) (2014) 19.

[33] A. Gandhi, P. Dube, A. Karve, A. Kochut, L. Zhang, Adaptive, model-driven autoscaling for cloud applications., 2014.

[34] Q. Guo, Y. Li, T. Liu, K. Wang, G. Chen, X. Bao, W. Tang, Correlation-based performance analysis for full-system mapreduce optimization, in: Big Data, 2013 IEEE International Conference on, IEEE, 2013, pp. 753–761.

Danilo Ardagna is an Associate Professor at the Dipartimento di Elettronica Informazione and Bioingegneria at Politecnico di Milano, Milan, Italy. He received the Ph.D. degree in Computer Engineering from Politecnico di Milano in 2004. His work focuses on performance modeling of software systems and on the design, prototype and evaluation of optimization algorithms for resource management and planning of big data and cloud systems.

Cinzia Cappiello is Assistant Professor in computer engineering at the Politecnico di Milano (Italy) from which she holds a Ph.D. in Information Technology (2005). Her research interests mainly regard data and information quality aspects in big data, service-based and Web applications, Web services and sensor data management. On such topics, she published papers in international journals and conferences. Cinzia is Associate Editor of the ACM Journal of Data and Information Quality. She has been co-chair of the workshops "Quality in Databases" in conjunction with VLDB 2010, "Data and Information Quality" in conjunction with CAiSE 2005, "Quality in Web Engineering" in conjunction with ICWE 2010–2013, and of the tracks "Information Quality Management in Innovative IS" of MCIS 2012 and "Data and Information quality" of ECIS 2008.

Walter Samà is a former master degree student at Politecnico di Milano. He obtained his master degree in Computer Science in 2017 with a thesis on Data Quality assessment in Big Data environments.

Monica Vitali received the Ph.D. in Information Technology from the Politecnico di Milano, Italy, in 2014. She is currently research assistant at the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano. She is interested in the topics of adaptation and monitorability in clouds and data centers, in adaptive and self-adaptive systems and services, and in Machine Learning techniques for adaptation. She has been involved in several international research projects related to cloud resource management and big data for smart cities.

- Data Quality assessment is a key success point for applications using big data
- Data quality assessment in big data requires approximation
- Confidence enables to give hints on data quality without a complete analysis
- Confidence is sensitive to the data source and the DQ metrics considered
- Optimization is used to select the best configuration for assessing DQ