Data Quality on KDD: a Real-life Scenario

Roberto Boselli, Mirko Cesarini, Fabio Mercorio, and Mario Mezzanzanica

Department of Statistics and Quantitative Methods, CRISP Research Centre,
University of Milan-Bicocca, Milan, Italy {firstname.lastname}@unimib.it
Discussion Paper

Abstract. The growing diffusion of IT-based services generates a lot of data useful for supporting the activities of firms, organisations, and state agencies. In such a context, data quality tasks are frequently addressed using cleansing routines, often framed in the wider context of ETL processes (Extraction, Transformation, and Loading). The design of these cleansing routines often relies on the experience of domain-experts, and this makes the evaluation of the quality level achieved a relevant concern to ensure the believability of the analysed results.
In this paper we describe two model based techniques aimed at respectively evaluating the consistency of a dataset and at identifying the cleansing alternatives. The techniques have been applied on a real-world dataset derived from the Italian labour market domain, which we made publicly available to the community.

Keywords: Data Cleansing, Data Quality, Model Checking, Model based approach, Data Visualisation

1 Introduction and Motivation

Research on data quality is growing in importance in both industrial and academic communities, as it aims at deriving knowledge (and then value) from data. Nowadays, Information Systems generate a lot of data able to describe the dynamics of subjects behaviours or phenomena over time (a.k.a. *longitudinal data*), making the design of data cleansing activities a key element for guaranteeing the believability of the overall knowledge discovery process.

In this direction, the existence of a connection between time-related and weakly-structured data has been recently clarified as follows [1]. Namely, let $Y(t)$ be an ordered sequence of observed data e.g., subject data sampled at different time $t \in T$, the observed data $Y(t)$ are weakly-structured if and only if the trajectory of $Y(t)$ resembles a random walk on a graph [2]. In such a context graphs or tree formalisms, which are used to model *weakly-structured* data, are also deemed appropriate for data quality activities. The main idea of the research we summarise here is that graph-based exploration techniques (e.g., model-checking, AI Planning) can contribute in both (i) evaluating data quality and (ii) automatically identifying cleansing alternatives.

A Real-World Scenario: The Labour Market Domain [3]. According to the Italian law, every time an employer hires or dismisses an employee, or an employment contract is modified a *Compulsory Communication* - an event - is sent to a job registry[1]. The Italian public administration has developed an ICT infrastructure [3] generating an administrative archive useful for studying the labour market dynamics (see, e.g. [4]). Each mandatory communication is stored into a record which presents several relevant attributes: **e_id** and **w_id** are ids identifying respectively the communication and the person involved; **e_date** is the event occurrence date whilst **e_type** describes the event type occurring to the worker's career, the event types can be the *start*, *cessation* and *extension* of a working contract, and the *conversion* from a contract type to a different one; **c_flag** states whether the event is related to a full-time or a part-time contract while **c_type** describes the contract type with respect to the Italian law, here we consider the *limited* (fixed-term) and *unlimited* (unlimited-term) contracts; finally, **empr_id** uniquely identifies the employer involved.

A *communication* represents an event arriving from the external world (ordered with respect to *e_date* and grouped by *w_id*), whilst a career is a longitudinal data sequence whose consistency has to be evaluated. To this end, the consistency semantics has been derived from the Italian labour law and from the domain knowledge that allowed the definition of the following *constraints*.

c1: an employee cannot have further contracts if a full-time is active;
c2: an employee cannot have more than K part-time contracts (signed by different employers), in our context we shall assume $K = 2$;
c3: an *unlimited term* contract cannot be extended;
c4: a contract extension can change neither the contract type (*c_type*) nor the modality (c_flag), for instance a part-time and fixed-term contract cannot be turned into a full-time contract by an extension;
c5: a conversion requires either the *c_type* or the c_flag to be changed (or both).

A Career Example. Table 1(a) shows a career violating the consistency requirements described above. The reader might notice that event 05 makes the career inconsistent since three part-time jobs are not allowed at the same time. As argued in [5], the consistency requirements are usually defined on either (i) a single tuple, (ii) two tuples or (iii) a set of tuples. While the first two classes can be modelled through FDs (Functional Dependencies) and their variants, the latter class requires reasoning with a (finite but not bounded) set of data items over time as the case of Tab. 1(a).

To the best of our knowledge, FDs have not been exploited for verifying and cleansing longitudinal data sequences. Indeed, catching inconsistencies over longitudinal data sequence, as shown in Tab. 1(a) through FDs could be an awkward task as detailed below. Let $\mathcal{R} = (R_1, \ldots, R_l)$ be a schema relation of the data showed in Tab. 1(a), and let R be an instance of $\mathcal{R}$. Then, let $P_k = R_1 \bowtie_{cond} R_2 \ldots \bowtie_{cond} R_k$ be the k^{th} Self Join (where *cond is* $R_i.Event = R_{i+1}.Event + 1$) i.e., P is R joined k times with itself on the condition *cond*

[1]Here the terms "communication" and "event" are considered as synonymous.

which (as an effect) put subsequent tuples in a row (with respect to the Event attribute). For simplicity we assume Tab. 1(a) to report data on only one career [2].

According to our experience, the k^{th} self join allows one to express constraints on Tab. 1(a) data using functional dependencies, however this approach has several drawbacks: (1) an expensive join of k relations is required to check constraints on sequences of k elements; (2) the start and cessation of a Part Time (e.g. events 01 and 03) can happen arbitrarily many times in a career before the event 05 is met (where the consistency is detected), therefore there can not be a k value higher enough to surely catch the inconsistency described; (3) the higher is k value the more the set of possible sequence (variations) to be checked; (4) Functional Dependencies do not provide any hint on how to fix inconsistencies, as discussed in [6]. For these reasons, in our research we are investigating a different approach to manage the problem showed in Tab. 1(a).

Some Related Works. In the database area, a lot of works have been focusing on *constraint-based data repair* for identifying errors by exploiting FDs (Functional Dependencies), multivalued dependencies, join dependencies, and inclusion dependencies. Two very effective approaches based on FDs are *database repair* [7] and *consistent query answering* [8]. The former aims to find a *repair* whilst the latter tries to compute *consistent query answers* in response to a query, namely answers that are true in every repair of the given database, but the source data is not fixed. Unfortunately, finding consistent answers to aggregate queries is a NP-complete problem already using two (or more) FDs [8, 9]. Furthermore, the usefulness of formal systems in databases has been motivated by observing that FDs are only a fragment of the first-order logic [10] while in [6] it has been observed that FDs fall short of guiding one in correcting errors.

More recently, the NADEEF tool [5] has been developed for creating a unified framework able to merge the most used cleansing solutions by both academy and industry. As the authors state, the NADEEF tool focuses on consistency requirements usually defined on either a single or two tuples. It actually represent a promising approach, although it still has to be evaluated on real-life domains.

The paper is organised as follows: in Sec. 2 the Robust Data Quality Analysis and the Universal Cleansing Process are introduced, framed in the context of data quality and weakly-structured data. Then in Sec. 3 we discuss some results and outline the concluding remarks.

2 Data Quality on Weakly-Structured Data

Here we briefly summarise the key elements of our model-based approach for data analysis and cleansing. First, we present the Robust Data Quality Analysis [11] and its Multidimensional enhancement [12, 13], a model-based technique aimed

[2] The above schema can be modified to manage data of several careers in the same table and it can be enhanced to manage sequences shorter than k by using the left outer join instead of the inner join as well.

at analysing the consistency of longitudinal paths before and after a cleansing intervention. Then, we discuss how such a model-based approach has been used for identifying cleansing alternatives [14] generating the so-called *Universal Cleanser*. The latter has been recently formalised as a AI Planning problem and proposed to the AI community [15, 16]. Data quality is a domain and goal dependent concept usually defined as "fitness for use". In our context, we shall focus on the inconsistency data quality dimension, which refers to "the violation of semantic rules defined over a set of data items or database tuples" [17].

2.1 Data Consistency Check

The approach we presented at [12] aims at modelling a *consistent* subject's evolution over time, then the authors verify if the subject's data evolve conforming to the model.

Intuitively, let us consider an events sequence $\epsilon = e_1, e_2, \ldots, e_n$ modelling the working example of Tab.1(a). Each event e_i will contain a number of observation variables whose evaluation determines a snapshot of the subject's *state*[3] at time point i, namely s_i. Then, the evaluation of any further event e_{i+1} might change the value of one or more state variables of s_i, generating a new state s_{i+1}.

Definition 1 (Events Sequence). *Let $\mathcal{R} = (R_1, \ldots, R_l)$ be a schema relation of a database. Then,*
(i) An event *$e = (r_1, \ldots, r_m)$ is a* record *of the projection $(R_1, \ldots, R_m)$ over $\mathcal{Q} \subseteq \mathcal{R}$ with $m \leq l$ s.t. $r_1 \in R_1, \ldots, r_m \in R_m$;*
(ii) Let $\sim$ be a total order *relation over events, an* event sequence *is a $\sim$-ordered sequence of events $\epsilon = e_1, \ldots, e_k$.*
A Finite State Event Dataset *S_i is an event sequence while a* Finite State Event Database *is a db S whose content is $S = \bigcup_{i=1}^{k} S_i$ with $k \geq 1$.*

We encode the subjects' behaviour (the so-called *consistency model*) on a transition system which allows representing each subject's data sequence as a pathway on a graph. The transition systems can be viewed as a graph describing the consistent evolution of weakly-structured data. Indeed, the use of a sequence of events $\epsilon = e_1, e_2, \ldots, e_n$ as input actions of the transition system deterministically determines a path $\pi = s_1 e_1 \ \ldots \ s_n e_n s_{n+1}$ on it (i.e., a *trajectory*), where a state s_j is the state resulting after the application of event e_i on s_i. Then, the data verification problem can be expressed as a model checking problem, which basically aims at verifying that a property is satisfied in each system state by using efficient graph exploration techniques for visiting the state space.

Namely, a model checker generates a trajectory for each event sequence: if a violation has been found both the trajectory and the event that triggered the violation are returned, otherwise the event sequence is consistent. Generally speaking, such a consistency check can be formalised as follows.

[3]The term "state" here is considered in terms of a value assignment to a set of finite-domain state variables

Definition 2 (*ccheck*)**.** *Let $\epsilon = e_1, \ldots, e_n$ be a sequence of events according to Definition 1, then ccheck : $FSED \rightarrow \mathbb{N} \times \mathbb{N}$ returns the pair $< i, er >$ where:*

(i) i is the index of a minimal subsequence $\epsilon_i = e_1, \ldots, e_i$ such that ϵ_{i+1} is inconsistent while $\forall j : j \leq i \leq n$, the subsequences ϵ_j are consistent.

(ii) er is zero if ϵ_n is consistent, otherwise it is a natural number which uniquely describes the inconsistency error code of the sequence ϵ_{i+1}.

We formalised the *ccheck* function exploiting the planning as model-checking paradigm, then we used the UPMurphi tool [18, 19] for realising it.

2.2 The Multidimensional Robust Data Quality Analysis

The RDQA was conceived to evaluate the effectiveness of a black-box cleansing routine (*clr* hereafter) on a specific dataset by addressing questions like *"what is the degree of cleanliness achieved through clr? Does the clr introduce any error in the cleansed dataset? Which is the margin of improvement of clr (if any)?* The RDQA can be iteratively applied on several *clr* (improved) versions until a satisfactory data quality level is reached taking as input: (i) a consistency model of the data, (ii) the source database S, and (iii) its cleansed instance C. While the cleansing function is executed on each $S_i \in S$ generating the cleansed version $C_i \in C$, the *ccheck* is used to verify the consistency of each event sequence S_i and C_i. The output of a RDQA iteration is the Double Check Matrix (DCM), as shown in Tab. 2, which allows one to assign each (S_i, C_i) pair to a DCM cluster. As an example, the Cluster 1 gives the number of event sequences in which both *ccheck* and *clr* agreed that the original data were clean and no intervention was required. On the contrary, the Cluster 4 shows the number of sequences for which no error was found by *ccheck* on S_i but a cleansing intervention took place, producing an inconsistent result.

Although this approach does not guarantee the correctness of the data cleansing process, it helps making the process more robust with respect to data consistency as each RDQA iteration acts like a *bug hunter* on the cleansing function.

The Multidimensional RDQA. Basically, the RDQA uses the *ccheck* function to analyse the effectiveness of a cleansing routine *clr*. Furthermore, as the *ccheck* supports the identification of *error-codes*, the model-checking-based exploration can be used for generating a taxonomy of all the (bounded) inconsistencies that may affect the data, according to the consistency model.

This can provide to the experts insights about the overall data inconsistency distribution, and the most relevant inconsistency patterns in the data. To this end, each cluster of the DCM is enriched with a square matrix having $n + 1$ rows and columns, where n is the number of error-codes detected, as shown in Tab. 1(b): An element of the matrix $err_{i,j}$ is a number $k \in \mathbb{N}$ if and only if k distinct event sequences have presented the error-code i in the original dataset and the error-code j in the cleansed one[4].

[4]For completeness we added the *dummy* error-code zero to represent consistent items e.g., $err_{0,0}$ is the number of consistent sequences before and after the cleansing.

Table 1. (a) A Career example where *P.T.* is for Part Time. (b) Error-codes matrix definition of a single DCM entry

(a)

Event	Event Type	Employer-ID	Date
01	P.T. Start	Firm 1	12^{th}/01/2010
02	P.T. Start	Firm 2	31^{st}/03/2011
03	P.T. Cessation	Firm 1	15^{th}/06/2011
04	P.T. Start	Firm 3	1^{st}/10/2012
05	P.T. Start	Firm 4	1^{st}/06/2013
...	...	...	...

(b)

$$\forall l \in \{1, \ldots, 8\}\ Cluster^{l}_{n+1,n+1} = \begin{bmatrix} err_{0,0} & err_{0,1} & \cdots & err_{0,n} \\ err_{1,0} & err_{1,1} & \cdots & err_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ err_{n,0} & err_{n,1} & \cdots & err_{n,n} \end{bmatrix}$$

2.3 Universal Cleanser (UC)

Let us consider an inconsistent event sequence $\epsilon = e_1, \ldots, e_n$ where the corresponding trajectory $\pi = s_1 e_1 \ \ldots \ s_n e_n s_{n+1}$ presents an event e_i leading to an inconsistent state s_j when applied on a state s_i. Intuitively, a *cleansing event sequence* can be seen as an alternative trajectory on the graph leading the subject's state from s_i to a new state where the event e_i can be applied (satisfying the consistency rules). The alternative trajectory is achieved adding a sequence of generated events between e_i and e_{i+1}. More formally we have the following.

Definition 3 (Cleansing event sequence). *Let $\epsilon = e_1, \ldots, e_n$ be an event sequence according to Def. 1 and let ccheck be a consistency function according to Def. 2. Let us suppose that the ϵ sequence is inconsistent, then $ccheck(\epsilon)$ returns a pair $< i, er >$ with $i > 0$ and an error-code $err > 0$.*

A Cleansing event sequence *ϵ_c is a non empty sequence of events $\epsilon_c = c_1, \ldots, c_m$ able to cleanse the inconsistency identified by the error code err, namely the new event sequence $\epsilon' = e_1, \ldots, e_i, c_1, \ldots, c_m, e_{i+1}$ is consistent, i.e., $second\{ccheck(\epsilon')\} = 0$.*

The exploration approach introduce above, formalised in terms of both model-checking and AI Planning problems respectively [14, 15], has been used to produce a *Universal Cleanser* (*UC*) that collects the set C of the synthesised cleansing action sequences $\epsilon_c \in C$ for each error-code *err* identified. Once generated, it can be used for realising a model-based *clr* function that, as a benefit, allows the *automatic* identification of cleansing alternatives.

3 Some Results and Concluding Remarks

We applied the Multidimensional-RDQA for evaluating the cleansing procedures used on people careers data. The dataset consists of $1,791,282$ mandatory communications describing the careers of $200,000$ people (a subset of an Italian Region inhabitants) observed starting from the 1^{st} January 2004 to the 31^{st} December 2011. The output of this process is the DCM shown in Tab. 2. The clusters labelled with $(*)$ represent the job careers dropped by the cleansing function (in spite of their consistency) as they refer to workers living outside the investigated region and they are not in the scope of the analysis.
Due to the space limitations, we restrict ourselves in commenting the following.

Table 2. The Double Check Matrix computed on careers data.

Row Number	*Cluster*	*What ccheck function says about a sequence?*			*Careers Data*	
		Is S_i consistent?	$S_i \stackrel{?}{=} C_i$	Is C_i consistent?	#Careers	%
R1	C1	Y	Y	Y	64,625	32.3
R2	C2	Y	Y	N	0	0.00
R3	C3	Y	N	Y	3,190	1.6
R4	C4	Y	N	N	184	0.09
R5	*	Y	N	unknown	315	0.15
R6	C5	N	Y	Y	0	0.00
R7	C6	N	Y	N	1,054	0.52
R8	C7	N	N	Y	116,216	58.1
R9	C8	N	N	N	14,059	7.02
R10	*	N	N	unknown	357	0.17

(i) The consistency degree (C1+C4) of the source dataset before and after the cleansing intervention. Only about 34% of the careers are consistent and this motivates the need of cleansing activities.

(ii) The room for improvement (C3+C4+C6+C8) as cases where the cleansing intervention failed. These clusters account for 9.23% of the careers, providing a quantitative estimation about how the *clr* could be improved.

(iii) The quality improvement (C7-C4) achieved by the cleansing intervention accounts for 58% of the careers. The use of a model-based approach to evaluate the cleansing process makes more reliable this value and the results obtained can be considered a measure of the *clr* effectiveness.

(iv) The Action Points. The capability to identify error patterns affecting the data, their distribution and characteristics is a useful swiss army knife to be used during the development of cleansing routines. Indeed, one can discover a set of issues and their relevance by analysing the DCM and the error codes distribution. To this aim, we exploited a well-suited multidimensional visualisation technique, namely the *parallel-coordinates*[5].

(v) **Universal Cleanser.** We computed the *Universal Cleanser* of the Labour Market Domain collecting all the cleansing alternatives for the 342 different error-codes identified, as discussed in [14]. It is worth highlighting that, as a characteristic, the UC is (i) computed off-line only once on the consistency model; (ii) it is data-independent since it can be used to cleanse any dataset conforming to the consistency model and (iii) it is policy-dependent since the cleansed results may vary as the policy varies. We are actually working on the automatic generation of an *accurate* policy, on the basis of the dataset to be analysed. Finally, an anonymous version of the analysed dataset, the Multidimensional RDQA outcomes, the parallel-coord demo, and the UC have been made publicly available for download and demonstration (`http://goo.gl/BTdES`), so that the results we present can be assessed, shared, and compared with other techniques.

Concluding Remarks. The traditional development of cleansing routines is a resource consuming and error prone activity. The size of data to be cleansed, the complexity of the domain, and the continuous business rules evolution make the cleansing process a challenging task. Our results, actually used at the CRISP Re-

[5]Parallel coordinates allow one to represent an n-dimensional datum $(x_1, \ldots, x_n)$ as a polyline, by connecting each x_i point in n parallel y-axes.

search Centre[6], show that a model-based data analysis and cleansing approach can effectively support domain experts in both in evaluating the level of data quality achieved and in *automatically* identifying cleansing solutions.

References

[1] Holzinger, A.: On knowledge discovery and interactive intelligent visualization of biomedical data - challenges in human-computer interaction & biomedical informatics. In Helfert, M., Francalanci, C., Filipe, J., eds.: DATA, SciTePress (2012)

[2] Kapovich, I., Myasnikov, A., Schupp, P., Shpilrain, V.: Generic-case complexity, decision problems in group theory, and random walks. J ALGEBRA **264**(2) (2003)

[3] The Italian Ministry of Labour and Welfare: Annual report about the CO system, available at `http://goo.gl/XdALYd` (2012)

[4] Lovaglio, P.G., Mezzanzanica, M.: Classification of longitudinal career paths. Quality & Quantity **47**(2) (2013)

[5] Dallachiesa, M., Ebaid, A., Eldawy, A., Elmagarmid, A.K., Ilyas, I.F., Ouzzani, M., Tang, N.: Nadeef: a commodity data cleaning system. In: SIGMOD. (2013)

[6] Fan, W., Li, J., Ma, S., Tang, N., Yu, W.: Towards certain fixes with editing rules and master data. Proceedings of the VLDB Endowment **3**(1-2) (2010) 173–184

[7] Chomicki, J., Marcinkowski, J.: Minimal-change integrity maintenance using tuple deletions. Information and Computation **197**(1) (2005) 90–121

[8] Bertossi, L.: Consistent query answering in databases. ACM Sigmod Rec. **35**(2) (2006) 68–76

[9] Chomicki, J., Marcinkowski, J.: On the computational complexity of minimal-change integrity maintenance in relational databases. In: Inconsistency Tolerance. Springer (2005) 119–150

[10] Vardi, M.: Fundamentals of dependency theory. Trends in Theoretical Computer Science (1987) 171–224

[11] Mezzanzanica, M., Boselli, R., Cesarini, M., Mercorio, F.: Data quality through model checking techniques. In: IDA. Volume 7014 of LNCS., Springer (2011)

[12] Boselli, R., Cesarini, M., Mercorio, F., Mezzanzanica, M.: Inconsistency knowledge discovery for longitudinal data management: A model-based approach. In: SouthCHI13 special session on HCI-KDD, LNCS, vol. 7947, Springer (2013)

[13] Boselli, R., Cesarini, M., Mercorio, F., Mezzanzanica, M.: A policy-based cleansing and integration framework for labour and healthcare data. In: Knowledge Discovery and Data Mining, LNCS 8401, Springer (2014) 141–168

[14] Mezzanzanica, M., Boselli, R., Cesarini, M., Mercorio, F.: Automatic synthesis of data cleansing activities. In: DATA 2013 - Proceedings of the International Conference on Data Technologies and Applications, SciTePress (2013)

[15] Boselli, R., Mezzanzanica, M., Cesarini, M., Mercorio, F.: Planning meets data cleansing. In: ICAPS, AAAI press (2014)

[16] Boselli, R., Mezzanzanica, M., Cesarini, M., Mercorio, F.: Towards data cleansing via planning. Intelligenza Artificiale `dx.doi.org/10.3233/IA-140061` **8**(1) (2014)

[17] Batini, C., Scannapieco, M.: Data Quality: Concepts, Methodologies and Techniques. Data-Centric Systems and Applications. Springer (2006)

[18] Della Penna, G., Intrigila, B., Magazzeni, D., Mercorio, F.: UPMurphi: a tool for universal planning on PDDL+ problems. In: ICAPS, AAAI Press (2009) 106–113

[19] Della Penna, G., Magazzeni, D., Mercorio, F.: A universal planning system for hybrid domains. Applied Intelligence **36**(4) (2012) 932–959

[6] Interuniversity Research Centre on Public Services - http://www.crisp-org.it