Improving Data Cleansing Accuracy: A model-based Approach

Mario Mezzanzanica, Roberto Boselli, Mirko Cesarini, and Fabio Mercorio
Department of Statistics and Quantitative Methods - C.R.I.S.P. Research Centre, University of Milan-Bicocca, Milan, Italy
{firstname.lastname}@unimib.it

Keywords: Data and Information Quality; Data Cleansing; Data Accuracy; Weakly-Structured Data

Abstract: 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. Information Systems generate a lot of data useful for studying the dynamics of subjects' behaviours or phenomena over time, making the quality of data a crucial aspect for guaranteeing the believability of the overall knowledge discovery process. In such a scenario, data cleansing techniques, i.e., automatic methods to cleanse a dirty dataset, are paramount. However, when multiple cleansing alternatives are available a *policy* is required for choosing between them. The policy design task still relies on the experience of domain-experts, and this makes the automatic identification of accurate policies a significant issue. This paper extends the Universal Cleaning Process enabling the automatic generation of an accurate cleansing policy derived from the dataset to be analysed. The proposed approach has been implemented and tested on an on-line benchmark dataset, a real-world instance of the Labour Market Domain. Our preliminary results show that our approach would represent a contribution towards the generation of data-driven policy, reducing significantly the domain-experts intervention for policy specification. Finally, the generated results have been made publicly available for downloading.

1 INTRODUCTION

Nowadays, public and private organizations recognise the value of data as a key asset to deeply understand social, economic, and business phenomena and to improve competitiveness in a dynamic business environment, as pointed out in several works (Fox et al., 1994; Madnick et al., 2009; Batini et al., 2009). In such a scenario, the field of KDD - Knowledge Discovery in Databases (KDD) field (Fayyad et al., 1996) - has received a lot of attention from several research and practitioner communities as it basically focuses on the identification of relevant patterns in data.

Data quality improvement techniques have rapidly become an essential part of the KDD process as most researchers agree that the quality of data is frequently very poor, and this makes data quality improvement and analysis crucial tasks for guaranteeing the believability of the overall KDD process[1] (Holzinger et al., 2013b; Pasi et al., 2013a). In such a direction, the *data cleansing* (a.k.a. data cleaning) research area concerns with the identification of a set of domain-dependent activities able to cleanse a dirty database (with respect to some given quality dimensions).

Although a lot of techniques and tools are available for cleansing data (e.g., the ETL-based tools[2]), a quite relevant amount of data quality analysis and cleansing design still relies on the experience of domain-experts that have to specify ad-hoc data cleansing procedures (Rahm and Do, 2000).

Here, a relevant question is how to guarantee that cleansing procedures results are accurate, or (at least) it makes sense for the analysis purposes, making the selection of *accurate* cleansing procedures (w.r.t. the analysis purposes) a challenging task that may consider the user's feedback for being accomplished properly (Cong et al., 2007; Volkovs et al., 2014).

This paper draws on two distinct works of (Holzinger, 2012) and (Mezzanzanica et al., 2013). Basically, the former has clarified that weakly-structured data can be modelled as a random walk on a graph during the DATA 2012 keynote (Holzinger, 2012). This, in turns, has en-

[1] Here the term *believability* is intended as "the extent to which data are accepted or regarded as true, real and credible"(Wang and Strong, 1996)

[2] The ETL (Extract, Transform and Load) process focuses on extracting data from one or more source systems, cleansing, transforming, and then loading the data into a destination repository, frequently a Datawarehouse. The ETL covers the data preprocessing and transformation tasks in the KDD process (Fayyad et al., 1996).

couraged the use of graph-search algorithms for both verifying the quality of such data and identifying cleansing alternatives, as shown by the work of (Mezzanzanica et al., 2013) and recently expressed also in terms of AI Planning problem (Boselli et al., 2014c; Boselli et al., 2014a).

Here we show how the approach of (Mezzanzanica et al., 2013) for the automatic identification of cleansing activities can be enhanced by selecting the more *accurate* one on the basis of the dataset to be cleansed. The technique has been formalised and realised by using the UPMurphi tool (Della Penna et al., 2009; Mercorio, 2013). Furthermore, we report our experimental results on the basis of the on-line dataset benchmark provided by (Mezzanzanica et al., 2013), making them publicly available to the community.

2 Motivation and Contribution

The *longitudinal data* (i.e., repeated observations of a given subject, object or phenomena at distinct time points) have received much attention from several academic research communities among the time-related data, as they are well-suited to model many real-world instances, including labour and healthcare domains, see, e.g. (Hansen and Järvelin, 2005; Holzinger, 2012; Holzinger and Zupan, 2013; Prinzie and Van den Poel, 2011; Lovaglio and Mezzanzanica, 2013; Devaraj and Kohli, 2000).

In such a context, graphs or tree formalisms, which are exploited to model *weakly-structured* data, are deemed also appropriate to model the expected longitudinal data behaviour (i.e., how the data should evolve over time).

Indeed, a strong relationship exists between weakly-structured and time-related data. 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 (Kapovich et al., 2003; De Silva and Carlsson, 2004).

Then, the Universal Cleansing framework introduced in (Mezzanzanica et al., 2013) can be used as a model-based approach to identify all the alternative cleansing actions. Namely, a graph modelling a weakly-structured data(set) can be used to derive a Universal Cleanser i.e., a taxonomy of possible errors (the term error is used in the sense of inconsistent data) and provides the set of possible corrections. The adjective *universal* would mean that (the universe of) every feasible inconsistency and cleansing action is identified with respect to the given model.

Table 1: An example of data describing the working career of a person. *P.T.* is for part-time, *F.T.* is for full-time.

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

The Universal Cleanser is presented as a foundation upon which data cleansing procedures can be quickly built. Indeed, the final step toward the creation of an effective data cleanser requires the identification of a policy driving the choice of which correction has to be applied when an error can be corrected in several ways.

Here we show how a policy can be inferred from the dataset to be analysed, by evaluating the possible cleansing actions over an artificially soiled dataset using an accuracy like metric. This, in turn, enhances the Universal Cleansing framework since the cleansing procedures can be automatically generated, thus avoiding domain-experts to manually specify the policy to be used.

In this regard, the term accuracy is usually related to the distance between a value v and a value v' which is considered correct or true (Scannapieco et al., 2005). As one might imagine, accessing the data *correct* value can be a challenging task, therefore such a value is often replaced by an approximation. Our idea is to exploit the part of the data evaluated as correct (thus accurate) for deriving a cleansing policy for cleansing the inconsistent part.

The following example should help in clarifying the matter.

Motivating Example. Some interesting information about the population can be derived using the labour administrative archive, an archive managed by a European Country Public Administration that records job start, cessation, conversion and extension events. The data are collected for administrative purposes and used by different public administrations.

An example of the labour archive content is reported in Table 1, a data quality examination on the labour data is shortly introduced. Data consistency rules can be inferred by the country law and common practice: an employee having one full-time contract can have no part-time contracts at the same time, an employee can't have more than k part-time contract at the same time (we assume $k = 2$). This consistency behaviour can be modelled through the graph shown in Fig. 1. Consistent data can be described by a path on the graph, while a path does not exist for inconsistent data.

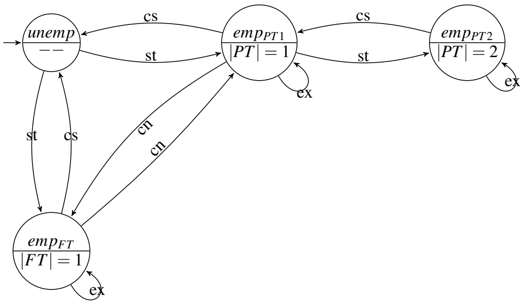


Figure 1: A graph representing the dynamics of a job career where $st = start$, $cs = cessation$, $cn = conversion$, and $ex = extension$. The PT and FT variables count respectively the number of part-time and the number of full-time jobs.

The reader would notice that the working history reported in Table 1 is inconsistent with respect to the semantic just introduced: a full-time Cessation is missing for the contract started by *event 03*. Looking at the graph, the worker was in node emp_{FT1} when *event 04* was received, however, *event 04* brings nowhere from node emp_{FT1}. A typical cleansing approach would add a full-time cessation event in a date between *event 03* and *event 04*. This would allow one to reach the node *unemp* through the added event and then the node emp_{PT1} via *event 04*, and this would make the sequence consistent with respect to the given constraints. However several other interventions could be performed e.g., to add a conversion event from full-time to part-time between *event 03* and *event 04* (reaching node emp_{PT1}). Notice that, in such a case, although the two distinct cleansing actions would make the career consistent, the former would create a career having one single part-time contract active while the second would make active two distinct part-time contracts. The question is: how to choose between these two alternatives? Indeed, although the last option may be unusual, a domain expert should take into account such a question since choosing one solution at the expense of the other may generate inconsistencies in the future, and vice-versa.

The comparison between archive contents and real data is often either unfeasible or very expensive (e.g. lack of alternative data sources, cost for collecting the real data, etc.). Then the development of cleansing procedures based on business rules still represent the most adopted solution by industry although the task of designing, implementing, and maintaining cleansing procedures requires a huge effort and is an error prone activity.

In such a scenario, one might look at the *consistent* careers of the dataset where similar situations occurred in order to derive a criterion, namely a policy, for selecting the most accurate cleansing option with respect to the dataset to be cleansed.

Contribution. Here we support the idea that an approach combining model driven cleansing (which identifies all the possible cleansing interventions based on a model of data evolution) and a cleansing policy (derived from the data and driving the selection of cleansing alternatives) can strengthen the effectiveness of the KDD process, by providing a more reliable cleansed dataset to data mining and analysis processes. Indeed, according to the whole KDD process, the data cleansing activities should be performed *after* the preprocessing task but *before* starting the mining activities by using formal and reliable techniques so that this would contribute in guaranteeing the believability of the overall knowledge process (Redman, 2013; Sadiq, 2013; Fisher et al., 2012).

This paper contributes to the problem of how to select a cleansing policy for cleansing data. To this end, we formalise and realise a framework, built on top of the Universal Cleansing approach we provided in (Mezzanzanica et al., 2013), that allows one:

- to model complex data quality constraints over longitudinal data, that represents a challenging issue. Indeed, as (Dallachiesa et al., 2013) argue, 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 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 longitudinal data, and this makes the exploration-based techniques a good candidate for that task;
- to automatically generate *accurate* cleansing policies with respect to the dataset to be analysed. Specifically, an accuracy based distance is used to identify the policy that can better restore to its original version an artificially soiled dataset. The policy, once identified, can be used to cleanse original inconsistent datasets. Clearly, several policy selection metrics can be used and consequently the analysts can evaluate several cleansing opportunities;
- to apply the proposed framework on a real-life benchmark dataset modelling the Italian Mandatory Communication domain (The Italian Ministry of Labour and Welfare, 2012) actually used

at CRISP Research Centre, then providing our results publicly available for downloading[3].

The outline of this paper is as follows. Sec. 3 outlines the Universal Cleansing Framework, Sec. 4 describes the automatic policy identification process, Sec. 5 shows the experimental results, Sec. 6 surveys the related works, and finally Sec. 7 draws the conclusions and outlines the future work.

3 UNIVERSAL CLEANSING

Here we briefly summarise the key elements of the framework presented by (Mezzanzanica et al., 2013) that allows one to formalise, evaluate, and cleanse longitudinal data sequences. The authors 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" (Batini and Scannapieco, 2006).

Intuitively, let us consider an events sequence $\varepsilon = e_1, e_2, \ldots, e_n$ modelling the working example of Tab.1. Each event e_i will contain a number of observation variables whose evaluation determines a snapshot of the subject's *state*[4] 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} in such a case. More formally we have the following.

Definition 1 (Events Sequence)**.** *Let $\mathcal{R} = (R_1, \ldots, R_l)$ be the schema of a database relation. 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 $\varepsilon = e_1, \ldots, e_k$.*

A Finite State Event Dataset *(FSED) S_i is an event sequence while a* Finite State Event Database *(FSEDB) is a database S whose content is $S = \bigcup_{i=1}^{k} S_i$ where $k \geq 1$.*

3.1 Data Consistency Check

Basically, the approach of (Mezzanzanica et al., 2013) aims at modelling a *consistent* subject's evolution over time according to a set of consistency requirements. Then, for a given subject, the authors verify if the subject's data evolve conforming to the model.

To this end the subjects' behaviour is encoded on a transition system (the so-called *consistency model*) so that each subject's data sequence can be represented 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 $\varepsilon = e_1, e_2, \ldots, e_n$ 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. This allows authors to cast the data consistency verification problem into a model checking problem, which is well-suited for performing efficient graph explorations.

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.

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

1. *i is the index of a minimal subsequence $\varepsilon_i = e_1, \ldots, e_i$ such that ε_{i+1} is inconsistent while $\forall j : j \leq i \leq n$, the subsequences ε_j are consistent.*
2. *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}.*

By abuse of notation, we denote as $first\{ccheck(S)\}$ the index i of the pair whilst $second\{ccheck(S)\}$ denotes the element er.

The authors implemented the *ccheck* function through model-checking based techniques and applied this approach for realising the (Multidimensional) Robust Data Quality Analysis (Boselli et al., 2013).

For the sake of completeness, we remark that the consistency requirements over longitudinal data described in (Mezzanzanica et al., 2013) could be expressed by means of FDs. Specifically, to the best of our knowledge, one might proceed as follows. Let us consider the career shown in Table 1 and let $\mathcal{R} = (R_1, \ldots, R_l)$ be a schema relation for these data. Moreover, let R be an instance of $\mathcal{R}$ and 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* which (as an effect) put subsequent tuples in a row (with respect to the Event attribute). For simplicity we assume Table1 to report data on only one career [5].

[3] Available at http://goo.gl/yy2fw3

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

[5] The above schema can be modified to manage data of

The k^{th} self join allows one to express constraints on Table 1 data using functional dependencies although this presents several drawbacks that we sketch as follows: (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 02) can occur arbitrarily many times in a career before the event 04 is met (where the inconsistency is detected), therefore there can not be a value of k higher enough to surely catch the inconsistency described; (3) the higher the value of k, 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 (Fan et al., 2010).

3.2 Universal Cleanser (UC)

When an inconsistency has been caught, the framework performs a further graph exploration for generating corrective events able to restore the consistency properties.

To clarify the matter, let us consider again an inconsistent event sequence $\varepsilon = e_1, \ldots, e_n$ as 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. What does "cleansing" such an event sequence mean? 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 (without violating the consistency rules). In other words, a *cleansing event sequence* for an inconsistent event e_i is a sequence of *generated* events that, starting from s_i, makes the subject's state able to reach a new state on which the event e_i can be applied resulting in a consistent state (i.e., a state that does not violate any consistency property). For example, considering the inconsistency described in Sec. 1 and the graph shown in Fig. 1, such cleansing action sequences are paths from the node emp_{FT} to the node emp_{PT1} or to the node *unemp*. For our purposes, we can formalise this concept as follows.

Definition 3 (Cleansing event sequence)**.** *Let $\varepsilon = e_1, \ldots, e_n$ 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(ε) returns a pair $< i, err >$ with $i > 0$ and an error-code $err > 0$.

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

Then, the *Universal Cleanser UC* can be seen as a collection that, for each error-code *err* returns the set C of the synthesised cleansing action sequences $\varepsilon_c \in C$. According to the author of (Mezzanzanica et al., 2013), the UC can be synthesised and used for cleansing a dirty dataset as shown in Figure 2(a).

4 AUTOMATIC POLICY IDENTIFICATION

The Universal Cleansing framework proposed by (Mezzanzanica et al., 2013) has been considered within this work since it presents some relevant characteristics useful for our purposes, namely:

1. it is computed off-line only once on the consistency model;
2. it is *data-independent* since, once the UC has been synthesised, it can be used to cleanse *any* dataset conforming to the consistency model;
3. it is *policy-dependent* since the cleansed results may vary as the policy varies.

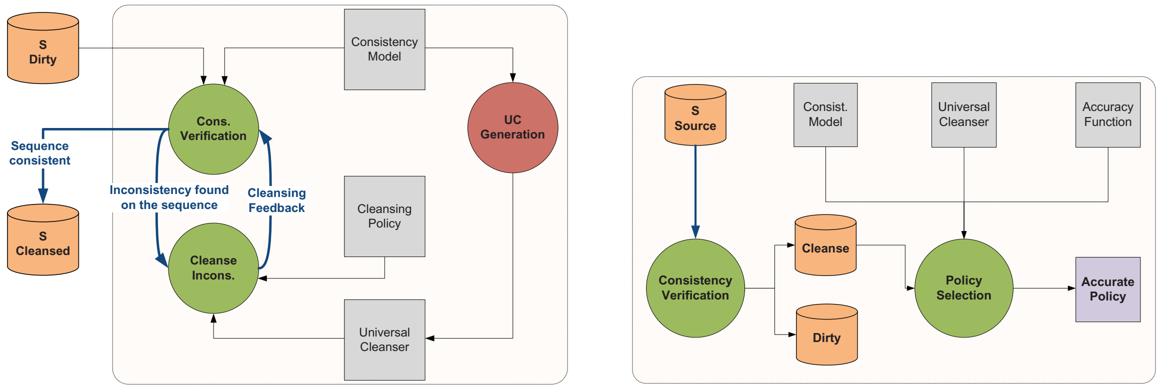
Focusing on (3), when several cleansing interventions are available for a given inconsistency, a *policy* (i.e., a criterion) is required for identifying which one has to be applied in the case.

The Universal Cleansing framework requires a policy to drive the data cleansing activities, and the policy identification task usually depends on the analysis purposes and the dataset characteristics as well. For these reasons, here we aim at automatically identifying an optimal cleansing policy, so that the data cleansing process (reported in Fig.2(a)) can be applied straightforward.

Clearly, a policy can be *optimal* if an optimality criterion is defined. This concept can be formalised according to the proposed framework as follows.

Definition 4 (Optimal Cleansing Policy)**.** *Let $\varepsilon = e_1, \ldots, e_n$ be an event sequence according to Def. 1 and let ccheck be a consistency function according to Def. 2.*
Let $err \in \mathbb{N}^+$ be an error-code identified on ε by the ccheck function and let $\mathcal{C}$ the set of all the cleansing action sequences $c_i \in \mathcal{C}$ able to cleanse the error code err. A cleansing policy $\mathcal{P} : \mathbb{N} \to 2^{\mathcal{C}}$ *maps the error code err to a list of cleansing action sequences $c_i \in \mathcal{C}$.*

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.



(a) The Universal Cleansing Frameworks, as presented by (Mezzanzanica et al., 2013). The UC is generated from the consistency model, then a policy is specified so that both the policy and the UC can be used for cleansing the data.

(b) The process for identifying an accurate cleansing policy. The data sequences considered consistent are fed to the policy selection process that identifies the optimal policy according to the specified accuracy function.

Figure 2: (a) The Universal Cleansing framework (taken from (Mezzanzanica et al., 2013)) and the Accurate Policy identification process as described in Sec.4

.

Finally, let $d : \mathbb{N}^{+} \times C \times C \to \mathbb{R}$ *be the* accuracy function, *an* optimal *cleansing policy* $\mathcal{P}^{*}$ *with respect to* d *assigns to each error-code the most accurate cleansing action sequence* c^{*}*, namely* $\forall err \in UC, \forall c_i \in \mathcal{C}, \exists c^{accurate}$ *such that* $d(err, c^{*}, c^{accurate}) \leq d(err, c_i, c^{accurate})$.

In Def. 4 the accuracy function basically evaluates the distance between a proposed corrective sequence (for a given error-code) against the corrective actions considered as accurate (i.e., the corrective sequence that transforms a wrong data to its real value). Unfortunately, the real value is hard to be accessed, as discussed above, therefore a proxy, namely a value considered as consistent according to the domain rules is frequently used.

In order to automatically identify a policy, we partition the source dataset into the consistent and inconsistent subsets respectively. Then, the consistent event sequences are used for identifying an accurate policy so as to cleanse the inconsistent subset. Basically, this can be accomplished through the following steps, whose pseudo-code is provided in Procedures 1, 2 and 3.

COMPUTE TIDY DATASET. Use the *ccheck* function for partitioning the source dataset S into S^{tidy} and S^{dirty} which represent the consistent and inconsistent subsets of S respectively;

ACCURATE POLICY IDENTIFICATION. For all ε from S^{tidy} (lines 2-12) and for all event $e_i \in \varepsilon$ (lines 3-11), generate a new sequence ε' obtained from ε by removing the event e_i (line 5) and verify the consistency on ε' (line 6). If ε' is inconsistent then use the resulting error-code *err* for retrieving the set of cleansing action sequences C from the Universal Cleanser (line 7);

SYNTHESISE POLICY. Compute the cleansing sequences minimising the the distance between c_i and the event deleted e_i (line 1). Once identified, enhance the UC properly. We recall that e_i is the most accurate correction (i.e. this is not a proxy) as it has been expressly deleted from the consistent sequence.

ACCURATE POLICY IDENTIFICATION. For each error-code *err* of the UC (line 1-3), compute the more accurate c_i and add it to the map $\mathcal{P}$ indexed by *err*.

Finally, for the sake of clarity, a graphical representation of this process has been given in Figure2(b).

Procedure 1 COMPUTE TIDY DATASET

```
Input: $S$ The Source Dataset
Output: $S^{tidy} \subseteq S$
1: $S^{tidy} \leftarrow \emptyset$
2: for all $\varepsilon = e_1, \ldots, e_n \in S$ do
3:     $< i, err > \leftarrow ccheck(\varepsilon)$
4:     if $err = 0$ then
5:         $S^{tidy} \leftarrow S^{tidy} \cup \{\varepsilon\}$
6:     end if
7: end for
8: return $S^{tidy}$
```

5 EXPERIMENTAL RESULTS

Procedure 2 ACCURATE POLICY IDENTIFICATION

```
Input: Consistency Model,
    $UC$ Universal Cleanser,
    $ccheck$ Consistency Function,
    $d$ Accuracy function,
    $S$ The Source Dataset, the FSEDB according to Def.1
Output: Cleansing Policy $\mathcal{P}$
$S^{validation} \leftarrow \emptyset$ //Will be used in Proc.4
$S^{tidy} \leftarrow$COMPUTE TIDY DATASET$(S)$
for all $\varepsilon = e_1, \dots, e_n \in S^{tidy}$ do
    for all $e_i \in \varepsilon$ do
        $\varepsilon'_i \leftarrow$ remove_event_from$(\varepsilon, e_i)$
        $< i, err > \leftarrow ccheck(\varepsilon'_i)$ //check $\varepsilon'_i$ consistency
        if $err \neq 0$ then
            SYNTHESISE POLICY$(UC[err])$
            $S^{validation} \leftarrow S^{validation} \cup \{\varepsilon'_i\}$
        end if
    end for
end for
for all $err \in UC$ do
    $\mathcal{P}[err] \leftarrow \min_{c_i \in UC[err]}(UC[err][c_i])$
end for
return $\mathcal{P}$
```

Procedure 3 SYNTHESISE POLICY

```
Input: $UC[err]$ The Set of Cleansing Sequences for $err$
$c_{min} \leftarrow \min_{c_i \in UC[err]} d(err, c_i, e_i)$
$UC[err][c_{min}] \leftarrow UC[err][c_{min}] + 1$
```

In this section the policy generation process described in Sec. 4 has been used for deriving a policy based upon the on-line benchmark domain provided by (Mezzanzanica et al., 2013). In Sec. 5.1 the domain model is described while in Sec. 5.2 some preliminary results are outlined.

5.1 The Labour Market Dataset

The scenario we are presenting focuses on the Italian labour market domain studied by statisticians, economists and computer scientists at the CRISP Research Centre.

According to the Italian law, every time an employer hires or dismisses an employee, or an employment contract is modified (e.g. from part-time to full-time), a *Compulsory Communication* - an event - is sent to a job registry. The Italian public administration has developed an ICT infrastructure (The Italian Ministry of Labour and Welfare, 2012) generating an administrative archive useful for studying the labour market dynamics (see, e.g.(Lovaglio and Mezzanzanica, 2013)). Each mandatory communication is stored into a record which presents several relevant attributes: **e_id** and **w_id** are ids identifying the communication and the person involved respectively; **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 as follows.

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).

5.2 Accurate Policy Generation Results

The process depicted in Fig.2(b) was realised as described in Sec.4 by using the following as input.

The Consistency Model was specified using the UPMurphi language and according to the consistency constraints specified in Sec. 5.1;

The on-line repository provided by (Mezzanzanica et al., 2013) and containing: (i) an XML dataset composed of $1,248,814$ records describing the career of $214,429$ people. Such events have been observed between 1^{st} January 2000 and 31^{st} December 2010; and (ii) the Universal Cleanser of the Labour Market Domain collecting all the cleansing alternatives for 342 different error-codes.

An Accuracy Function usually intended as the distance between a value v and a value v' which is considered correct or true. As the consistency we model is defined over a *set* of data items rather than a single attribute, the accuracy function should deal with all the event attribute values, as we clarified in the motivating example. To this end, the *edit distance* over the merged attribute values was used. Specifically, it is well-suited for comparing two strings by measuring the minimum number of operations needed to transform one string into another, taking into account insertion, deletion, substitution as well as the transposition of two adjacent characters. Notice that the edit distance is often used for measuring similarities between phrases, then it is successfully applied in natural language processing, bio-informatics, etc. Thus, as a mandatory communication is composed by strings, the edit distance metric represents a good candidate for realising the *accuracy* function.

The Proc. 1, 2, and 3 of Sec.4 have been implemented through C++ while the *ccheck* function basically calls the UPMurphi planner searching for an inconsistency. The process required about 2 hours running on a x64 Linux-machine, equipped with 6GB of RAM and connected to a MySQL DMBS for retrieving the data.

The output of such a process is an *accurate policy* according to the Def. 4, which we visualise in Fig 3.

The x axis reports the error-codes caught by applying Procedure 2 on S^{tidy} while two y-axes are provided. The rightmost reports how many times an error-code has been encountered during the policy generation process using a logarithmic scale (the black triangle) while the leftmost shows the percentage of successfully cleansing action applications.

Here, some results are commented as follows.

Result Comments. The error-code numbering was performed using a proximity criterion, the similar the inconsistencies the closer the error codes. The coloured bars refer to the leftmost axis showing the percentage of cases that have been successfully corrected using a specific cleansing action. For this dataset, 17 cleansing actions of the universal cleanser were applied at least once.

To give a few examples, we discovered that the most adopted cleansing intervention is $C11$ as it has been applied up to $49,358$ times for a situation like the following: a person having only an active part-time contract with a given company A received a cessation event for another part-time contract with another company B. This clearly represents an inconsistency for which the most accurate correction event was $C11$, namely the introduction of a single corrective event $c_i = (start, PT,\text{Limited},\text{Company } B)$.

On the contrary, the error-code 84 represents an unemployed subject that starts his career receiving a conversion event of a (non-existing) working contract, namely $e_1 = (conversion, FT,\ Unlimited,\text{Company } A)$. Although one might easily argue that a *start* of a contract with company A is the only thing capable of making career consistent, it might not be so easy to identify the most accurate attribute values of the event to be added (e.g., the contract type). Indeed, several cleansing alternatives were identified as accurate during the process, namely to consider (C11) a full-time unlimited contract; (C15) a part-time unlimited contract and (C13) a part-time limited one.

Another case worth of mentioning is the error-code 31: a person is working for two companies A and B and a communication is received about the start of a third part-time contract with Company C. Here, two cleansing alternatives are available: to close the contract either with A (C2) or with B (C3). This is a classical example that reveals the importance of an accurate policy identification, as one must define (and motivate) why a correction has been preferred over another. Thus, such a kind of result is quite important for domain-experts as it represents a criterion derived (automatically) on the basis of the dataset that they intend to analyse and cleanse.

Considering the results of Fig. 3, for some inconsistencies there is only a cleansing alternative (e.g. error-codes from 71 to 82). On the other hand, when more alternative cleansing are available, very frequently one of them greatly outperforms the others in terms of number of successfully cleansed cases. Moreover, we discovered that similar error-codes are successfully cleansed by the same cleansing actions (e.g. the error codes from 9 to 14, from 59 to 66). Generally speaking, the overall plot of Fig.2(b) provides useful information for the development of cleansing procedures.

Finally, it is worth noting that the edit distance value for every cleansed sequence was zero, and this confirms that the UC generated is really capable to identify effective cleansing actions.

5.3 Accuracy Estimation

Here, the *k-fold cross validation* technique (see, e.g. (Kohavi, 1995)) was used to evaluate the policy synthesised according to the framework proposed.

Generally speaking, the dataset S is split into k mutually exclusive subsets (the folds) $S_1, \ldots, S_k$ of approximatively equal size. Then, k evaluation steps are performed as follows: at each step t the set of events $S \setminus S_t$ is used to identify the cleansing policy $\mathcal{P}_t$, whilst the left out subset S_t is used for evaluating the cleansing accuracy achieved by $\mathcal{P}_t$. In each step, it is computed the ratio of the event sequences the cleanser has made equal to the original via $\mathcal{P}_t$ over the cardinality of all the cleansed event sequences, namely $|S_t|$. Then, an average of the k ratios is computed. These steps are described by the pseudo-code given in Proc. 4.

Procedure 4 K-FOLD CROSS VALIDATION

```
1:  $S \leftarrow S^{validation}$
2:  //Step1: generate folds
3:  for all $t \in \{1,\dots,k\}$ do
4:      $S_t \leftarrow$random_sampling$(S,k)$ //such that $\cup S_t = S$ and $\cap S_t = \emptyset$ and $|S_t| = k$
5:  end for
6:  //Step2: iterate on folds
7:  for all $t \in \{1,\dots,k\}$ do
8:      $match \leftarrow 0$
9:      $S' \leftarrow S \setminus S_t$
10:     //Step2.1: synthesise a new policy $\mathcal{P}_t$
11:     for all $\varepsilon \in S'$ do
12:         SYNTHESISE POLICY$(UC[err_\varepsilon])$
13:     end for
14:     for all $err \in UC$ do
15:         $\mathcal{P}_t[err] \leftarrow \min_{c_i \in UC[err]}(UC[err][c_i])$
16:     end for
17:     //Step2.2: evaluate policy $\mathcal{P}_t$ against $\mathcal{P}_t$
18:     for all $\varepsilon = e_1,\dots,e_n \in S'$ do
19:         $\varepsilon_{cleansed} \leftarrow$ apply_policy$(\mathcal{P}_t[err_\varepsilon],\varepsilon)$
20:         $\varepsilon_{tidy} \leftarrow$ retrieve_original_from$(S^{tidy},\varepsilon)$
21:         if $\varepsilon_{cleansed} = \varepsilon_{tidy}$ then
22:             $match \leftarrow match + 1$
23:         end if
24:     end for
25:     $\mathcal{M}_t \leftarrow \frac{match}{|S_t|}$
26: end for
27: //Step3: compute policy accuracy
28: $\mathcal{M}^k \leftarrow \frac{1}{k}\sum_{t=1}^{k}\mathcal{M}_t$
```

Step 1: Generate k folds by splitting the $S^{validation}$ dataset (lines 1-5). Notice that the $S^{validation}$ was computed previously by Procedure 1.

Step 2: For each t up to k, (i) iteratively generate a new policy[6] $\mathcal{P}_t$ by using only the dataset $S \setminus S_t$ as input (lines 7-16). (ii) Evaluate the instance of ε cleansed according to the policy $\mathcal{P}_t$, i.e. comprare $\varepsilon_{cleansed}$ with the original instance of ε as appears in S^{tidy}, namely ε_{tidy} (lines 17-26).

Step 3: Compute the overall $\mathcal{P}$ accuracy, i.e., $\mathcal{M}^k$ as the average of the accuracy values $\mathcal{M}_t$ reached by each fold iteration (line 28).

This method has the advantage of using all the available data for both estimating and evaluating the policy, during the k steps. Indeed, although $k = 10$ is usually considered as a good value in the common practice, we used a variable value of k ranging in $\{5,10,50,100\}$ so that different values of accuracy $\mathcal{M}^k$ can be analysed, as summarised in Tab. 2. The results confirm the high-degree of accuracy reached by the policy $\mathcal{P}$ synthesised in Sec.5.2. Indeed, during the iterations no single $\mathcal{M}^k$ value was lower than 99.4% and peaks of 100% were reached during some iterations.

Table 2: Values of $\mathcal{M}^k$ applying k-cross validation of Procedure 4.

k	5	10	50	100
$\mathcal{M}^k$	99.77%	99.78%	99.78%	99.78%
$\min(\mathcal{M}^k)$	99.75%	99.75%	99.48%	99.48%
$\max(\mathcal{M}^k)$	99.77%	99.80%	99.92%	100%
$|S_t|$	42,384	21,193	4,238	2,119
$|S|$	211,904			

6 RELATED WORK

The data quality analysis and improvement tasks have been the focus of a large body of research in different domains, that involve statisticians, mathematicians and computer scientists, working in close cooperation with application domain experts, each one focusing on its own perspective (Abello et al., 2002; Fisher et al., 2012). Computer scientists developed algorithms and tools to ensure data correctness by paying attention to the whole Knowledge Discovery process, from the collection or entry stage to data visualisation (Holzinger et al., 2013a; Ferreira de Oliveira and Levkowitz, 2003; Clemente et al., 2012; Fox et al., 1994; Boselli et al., 2014b). From a statistical perspective, *data imputation* (a.k.a. data editing) is performed to replace null values or, more in general, to address quality issues while preserving the collected data statistical parameters (Fellegi and Holt, 1976).

On the other hand, a common technique exploited by computer scientists for data cleansing is the *record linkage* (a.k.a. *object identification*, *record matching*, *merge-purge problem*), that aims at linking the data to a corresponding higher quality version and

[6] Notice that the creation of a new policy is not an expensive task as it does not require to invoke *ccheck*.

to compare them (Elmagarmid et al., 2007). However, when an alternative (and more trusted) data to be linked are not available, the most adopted solution (specially in the industry) is based on *Business Rules*, identified by domain experts for checking and cleansing dirty data. These rules are implemented in SQL or in other tool specific languages. The main drawback of this approach is that the design relies on the experience of domain experts; thus exploring and evaluating cleansing alternatives quickly become a very time-consuming task: each business rule has to be analysed and coded separately, then the overall solution still needs to be manually evaluated.

In the database area, many works focus on *constraint-based data repair* for identifying errors by exploiting FDs (Functional Dependencies) and their variants. Nevertheless, the usefulness of formal systems in databases has been motivated in (Vardi, 1987) by observing that FDs are only a fragment of the first-order logic used in formal methods while (Fan et al., 2010) observed that, even though FDs allow one to detect the presence of errors, they have a limited usefulness since they fall short of guiding one in correcting the errors.

Two more relevant approaches based on FDs are *database repair* (Chomicki and Marcinkowski, 2005a) and *consistent query answering* (Bertossi, 2006). The former aims to find a *repair*, namely a database instance that satisfies integrity constraints and minimally differs from the original (maybe inconsistent) one. The latter approach tries to compute *consistent query answers* in response to a query i.e., 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 (Bertossi, 2006; Chomicki and Marcinkowski, 2005b). To mitigate this problem, recently a number of works have exploited heuristics to find a database repair, as (Yakout et al., 2013; Cong et al., 2007; Kolahi and Lakshmanan, 2009). They seem to be very promising approaches, even though their effectiveness has not been evaluated on real-world domains.

More recently, the NADEEF (Dallachiesa et al., 2013) tool has been developed in order to create a unified framework able to merge the most used cleansing solutions by both academy and industry. In our opinion, NADEEF gives an important contribution to data cleansing also providing an exhaustive overview about the most recent (and efficient) solutions for cleansing data. Indeed, as the authors remark, consistency requirements are usually defined on either a single tuple, two tuples or a set of tuples. The first two classes are enough for covering a wide spectrum of basic data quality requirements for which FD-based approaches are well-suited. However, the latter class of quality constraints (that NADEEF does not take into account according to its authors) requires reasoning with a (finite but not bounded) set of data items over time as the case of longitudinal data, and this makes the exploration-based technique a good candidate for that task.

Finally, the work of (Cong et al., 2007), to the best of our knowledge, can be considered a milestone in the identification of accurate data repairs. Basically, they propose (i) a heuristic algorithm for computing a data repair satisfying a set of constraints expressed through *conditional* functional dependencies and (ii) a method for evaluating the accuracy of the repair. Compared with our work their approach differs mainly for two aspects. First, they use conditional FDs for expressing constraints as for NADEEF, while we consider constraints expressed over more than two tuples at a time. Second, their approach for evaluating the accuracy of a repair is based upon the user feedback. Indeed, to steam the user effort only a *sample* of the repair - rather than the entire repair - is proposed to the user according to authors. Then a confidence interval is computed for guaranteeing a precision higher than a specified threshold. On the contrary, our approach reduces the user effort in the synthesis phase exploiting a graph search for computing accurate cleansing alternatives.

7 CONCLUDING REMARKS AND FURTHER DIRECTIONS

In this paper the Universal Cleansing framework was extended in order to enable the *automatic* identification of an accurate policy (on the basis of the dataset to be analysed) used for choosing the cleansing intervention to be applied when there are several available. This represents the final step to achieve the selection of accurate cleansing procedures. To this end, we modelled the consistency behaviour of the data through UPMurphi, a model checking based tool used for verifying the data consistency. Then, the whole consistent portion of the source dataset was made inconsistent so that several cleansing interventions could be evaluated.

As a result, a *data-driven* policy was obtained summarizing only the cleansing actions that minimise the accuracy function for a given error-code. Here, we used the well-known ”edit distance” metric for measuring the accuracy between the real data and the proposed cleansing action, although our approach supports the use of different metrics.

The main benefit of our approach is in the automatic policy generation, as it (i) speeds up the development of cleansing procedures, (ii) provides insights about the dataset to be cleansed to domain-experts and (iii) dramatically reduces the human effort required for identifying an accurate policy.

Finally, our approach has been successfully tested for generating an accurate policy over a real (weakly-structured) dataset describing people working careers and tested according to the well-known k-fold cross validation method.

As a further step, we have been working toward evaluating the effectiveness of our approach on biomedical domain. Finally, we intend to deploy our policy on the system actually used at CRISP Research Centre, which is based on an ETL tool. This would represent a contribution toward the realisation of a hybrid approach that combines a mode-based policy generation with a business-rules-based cleansing approach.

REFERENCES

Abello, J., Pardalos, P. M., and Resende, M. G. (2002). *Handbook of massive data sets*, volume 4. Springer.

Batini, C., Cappiello, C., Francalanci, C., and Maurino, A. (2009). Methodologies for Data Quality Assessment and Improvement. *ACM Comput. Surv.*, 41:16:1–16:52.

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

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

Boselli, R., Cesarini, M., Mercorio, F., and Mezzanzanica, M. (2013). Inconsistency knowledge discovery for longitudinal data management: A model-based approach. In *SouthCHI13 special session on Human-Computer Interaction & Knowledge Discovery, LNCS, vol. 7947*. Springer.

Boselli, R., Cesarini, M., Mercorio, F., and Mezzanzanica, M. (2014a). Planning meets data cleansing. In *The 24th International Conference on Automated Planning and Scheduling (ICAPS)*, pages 439–443. AAAI.

Boselli, R., Cesarini, M., Mercorio, F., and Mezzanzanica, M. (2014b). A policy-based cleansing and integration framework for labour and healthcare data. In Holzinger, A. and Igor, J., editors, *Interactive Knowledge Discovery and Data Mining in Biomedical Informatics*, volume 8401 of *LNCS*, pages 141–168. Springer.

Boselli, R., Cesarini, M., Mercorio, F., and Mezzanzanica, M. (2014c). Towards data cleansing via planning. *Intelligenza Artificiale*, 8(1):57–69.

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

Chomicki, J. and Marcinkowski, J. (2005b). On the computational complexity of minimal-change integrity maintenance in relational databases. In *Inconsistency Tolerance*, pages 119–150. Springer.

Clemente, P., Kaba, B., Rouzaud-Cornabas, J., Alexandre, M., and Aujay, G. (2012). Sptrack: Visual analysis of information flows within selinux policies and attack logs. In *AMT Special Session on Human-Computer Interaction and Knowledge Discovery*, volume 7669 of *LNCS*, pages 596–605. Springer.

Cong, G., Fan, W., Geerts, F., Jia, X., and Ma, S. (2007). Improving data quality: Consistency and accuracy. In *Proceedings of the 33rd international conference on Very large data bases*, pages 315–326. VLDB Endowment.

Dallachiesa, M., Ebaid, A., Eldawy, A., Elmagarmid, A. K., Ilyas, I. F., Ouzzani, M., and Tang, N. (2013). Nadeef: a commodity data cleaning system. In Ross, K. A., Srivastava, D., and Papadias, D., editors, *SIGMOD Conference*, pages 541–552. ACM.

De Silva, V. and Carlsson, G. (2004). Topological estimation using witness complexes. In *Proceedings of the First Eurographics conference on Point-Based Graphics*, pages 157–166. Eurographics Association.

Della Penna, G., Intrigila, B., Magazzeni, D., and Mercorio, F. (2009). UPMurphi: a tool for universal planning on PDDL+ problems. In *Proceeding of the 19th International Conference on Automated Planning and Scheduling (ICAPS) 2009*, pages 106–113. AAAI Press.

Devaraj, S. and Kohli, R. (2000). Information technology payoff in the health-care industry: a longitudinal study. *Journal of Management Information Systems*, 16(4):41–68.

Elmagarmid, A. K., Ipeirotis, P. G., and Verykios, V. S. (2007). Duplicate record detection: A survey. *Knowledge and Data Engineering, IEEE Transactions on*, 19(1):1–16.

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

Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34.

Fellegi, I. P. and Holt, D. (1976). A systematic approach to automatic edit and imputation. *Journal of the American Statistical association*, 71(353):17–35.

Ferreira de Oliveira, M. C. and Levkowitz, H. (2003). From visual data exploration to visual data mining: A survey. *IEEE Trans. Vis. Comput. Graph.*, 9(3):378–394.

Fisher, C., Lauría, E., Chengalur-Smith, S., and Wang, R. (2012). *Introduction to information quality*. AuthorHouse.

Fox, C., Levitin, A., and Redman, T. (1994). The notion of data and its quality dimensions. *Information processing & management*, 30(1):9–19.

Hansen, P. and Järvelin, K. (2005). Collaborative information retrieval in an information-intensive domain. *Information Processing & Management*, 41(5):1101–1119.

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

Holzinger, A., Bruschi, M., and Eder, W. (2013a). On interactive data visualization of physiological low-cost-sensor data with focus on mental stress. In Cuzzocrea, A., Kittl, C., Simos, D. E., Weippl, E., and Xu, L., editors, *CD-ARES*, volume 8127 of *Lecture Notes in Computer Science*, pages 469–480. Springer.

Holzinger, A., Yildirim, P., Geier, M., and Simonic, K.-M. (2013b). Quality-based knowledge discovery from medical text on the web. In (Pasi et al., 2013b), pages 145–158.

Holzinger, A. and Zupan, M. (2013). Knodwat: A scientific framework application for testing knowledge discovery methods for the biomedical domain. *BMC Bioinformatics*, 14:191.

Kapovich, I., Myasnikov, A., Schupp, P., and Shpilrain, V. (2003). Generic-case complexity, decision problems in group theory, and random walks. *Journal of Algebra*, 264(2):665–694.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1137–1143, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Kolahi, S. and Lakshmanan, L. V. (2009). On approximating optimum repairs for functional dependency violations. In *Proceedings of the 12th International Conference on Database Theory*, pages 53–62. ACM.

Lovaglio, P. G. and Mezzanzanica, M. (2013). Classification of longitudinal career paths. *Quality & Quantity*, 47(2):989–1008.

Madnick, S. E., Wang, R. Y., Lee, Y. W., and Zhu, H. (2009). Overview and framework for data and information quality research. *J. Data and Information Quality*, 1(1):2:1–2:22.

Mercorio, F. (2013). Model checking for universal planning in deterministic and non-deterministic domains. *AI Commun.*, 26(2):257–259.

Mezzanzanica, M., Boselli, R., Cesarini, M., and Mercorio, F. (2013). Automatic synthesis of data cleansing activities. In Helfert, M. and Francalanci, C., editors, *The 2nd International Conference on Data Management Technologies and Applications (DATA)*, pages 138 – 149. Scitepress.

Pasi, G., Bordogna, G., and Jain, L. C. (2013a). An introduction to quality issues in the management of web information. In (Pasi et al., 2013b), pages 1–3.

Pasi, G., Bordogna, G., and Jain, L. C., editors (2013b). *Quality Issues in the Management of Web Information*, volume 50 of *Intelligent Systems Reference Library*. Springer.

Prinzie, A. and Van den Poel, D. (2011). Modeling complex longitudinal consumer behavior with dynamic bayesian networks: an acquisition pattern analysis application. *Journal of Intelligent Information Systems*, 36(3):283–304.

Rahm, E. and Do, H. (2000). Data cleaning: Problems and current approaches. *IEEE Data Engineering Bulletin*, 23(4):3–13.

Redman, T. C. (2013). Data's credibility problem. *Harvard Business Review*, 91(12):84–+.

Sadiq, S. (2013). *Handbook of Data Quality*. Springer.

Scannapieco, M., Missier, P., and Batini, C. (2005). Data Quality at a Glance. *Datenbank-Spektrum*, 14:6–14.

The Italian Ministry of Labour and Welfare (2012). Annual report about the CO system, available at `http://www.cliclavoro.gov.it/Barometro-Del-Lavoro/Documents/Rapporto_CO/Executive_summary.pdf`.

Vardi, M. (1987). Fundamentals of dependency theory. *Trends in Theoretical Computer Science*, pages 171–224.

Volkovs, M., Chiang, F., Szlichta, J., and Miller, R. J. (2014). Continuous data cleaning. *ICDE (12 pages)*.

Wang, R. Y. and Strong, D. M. (1996). Beyond accuracy: What data quality means to data consumers. *J. of Management Information Systems*, 12(4):5–33.

Yakout, M., Berti-Équille, L., and Elmagarmid, A. K. (2013). Don't be scared: use scalable automatic repairing with maximal likelihood and bounded changes. In *Proceedings of the 2013 international conference on Management of data*, pages 553–564. ACM.

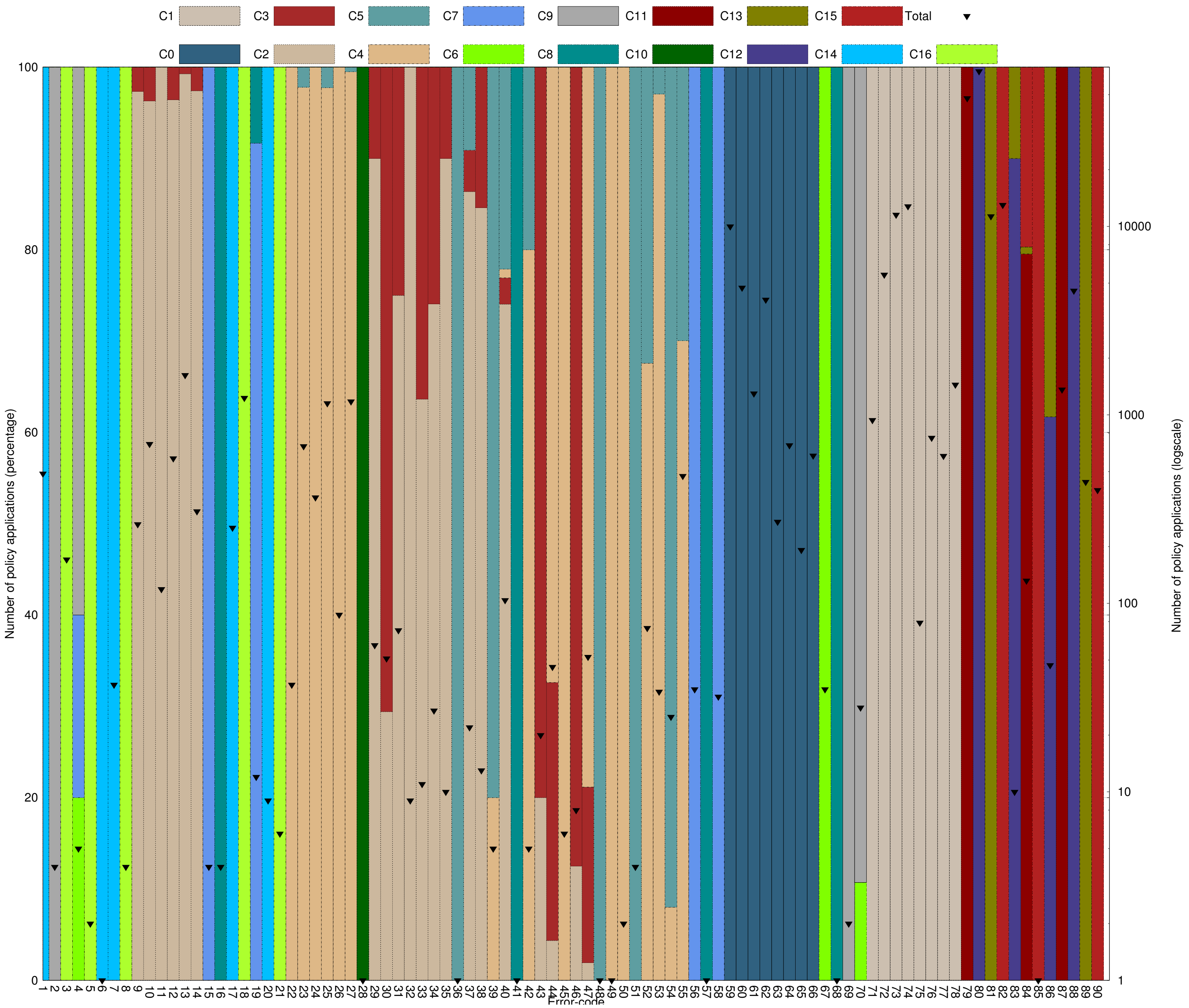


Figure 3: A plot showing the cleansing interventions successfully performed for each error code, during the policy selection process performed against the benchmark dataset. The *x* axis reports the 90 error-codes identified on the soiled dataset. The rightmost *y*-axis reports how many times an error-code was encountered during the policy generation process using a logarithmic scale (the black triangle) while the coloured bars refer to the leftmost axis showing the percentage of cases that were successfully corrected using a specific cleansing action.