Improving Data Quality by Leveraging Statistical Relational Learning

LARYSA VISENGERIYEVA, Technische Universität Berlin, Germany
ALAN AKBIK, IBM Research - Almaden, USA
MANOHAR KAUL, IIT Hyderabad, India

Digitally collected data suffers from many data quality issues, such as duplicate, incorrect, or incomplete data. A common approach for counteracting these issues is to formulate a set of data cleaning rules to identify and repair incorrect, duplicate and missing data. Data cleaning systems must be able to treat data quality rules holistically, to incorporate heterogeneous constraints within a single routine, and to automate data curation. We propose an approach to data cleaning based on statistical relational learning (SRL). We argue that a formalism - Markov logic - is a natural fit for modeling data quality rules. Our approach allows for the usage of probabilistic joint inference over interleaved data cleaning rules to improve data quality. Furthermore, it obliterates the need to specify the order of rule execution. We describe how data quality rules expressed as formulas in first-order logic directly translate into the predictive model in our SRL framework.

1. INTRODUCTION

Having access to high quality data is of great importance in data analysis. However, data in the real world is often considered *dirty*: it contains inaccurate, incomplete, inconsistent, duplicated, or stale values. A number of distinct data quality issues are known in the field of data quality management such as *data consistency*, *currency*, *accuracy*, *deduplication* and *information completeness* [Fan and Geerts 2012]. As previous work has observed, such data quality issues are detrimental to data analysis [Council 2013],[Fan and Geerts 2012] and cause huge costs to businesses [Eckerson 2002]. Therefore, improving data quality with respect to business and integrity constraints is a crucial component of data management. A common approach to increase data quality is to formulate a set of *data cleaning rules* that detect semantic errors by utilizing data dependencies [Fan and Geerts 2012], [Arasu et al. 2009], [Dallachiesa et al. 2013], [Geerts et al. 2013]. However, previous research identified a number of requirements and accompanying challenges, which are associated with creating such rule sets (c.f., Section 2):

Interleaved rules. First, while each such rule usually addresses one data quality issue individually, the individual rules as a whole typically *interact* [Fan and Geerts 2012], [Fan et al. 2014]. For instance, a rule that deletes duplicates might perform better after missing data has already been imputed, while, on the other hand, a rule that imputes missing data might perform better if duplicates have already been removed. Therefore, we argue to model data quality rules such as deduplication and missing value imputation *jointly*, rather than as separate processes. Second, rules in such a rule-set may need to be modeled *"soft"* and *"hard"* in order to balance constraints of different importance [Yakout et al. 2013], especially within a set of interacting rules.

Automation. Different execution orders of interleaved rules produce different results [Dallachiesa et al. 2013]. Imposing the difficult problem of manually specifying the execution order on the user conflicts with the automation principle of data curation systems [Stonebraker et al. 2013].

Author's email: L.Visengeriyeva larysa.visengeriyeva@tu-berlin.de;

DOI: 0000001.0000001

Usability and domain knowledge integration. Various languages and statistical approaches for data curation exist [Dallachiesa et al. 2013], [Chu et al. 2013], [Geerts et al. 2013]. However, there is a need for expressiveness and customization of the rules in order to integrate arbitrary constraints into data cleaning without having to specify complex user-defined functions.

In this paper, we present an approach to data cleaning based on statistical relational learning (SRL) [Getoor and Taskar 2007] and probabilistic inference (c.f., Section 3). SRL is a branch of machine learning that models joint distributions over relational data. Generally, data quality rules represent relationships between attributes in the database schema. These rules are mainly based on integrity constraints such as functional dependencies [Abiteboul et al. 1995], [Fan and Geerts 2012] on top of a database schema. We show how to translate such functional dependencies, expressed as first-order logic formulas, into probabilistic-logical languages, which allows us to reason over inconsistencies, duplicates or missing values in a probabilistic way (c.f., Section 3.1). During automatic data cleaning, the optimal order of rules execution is hardly achievable [Dallachiesa et al. 2013]. Therefore, we propose to use joint inference for the simultaneous rules execution.

The contributions of this paper are the following:

We propose how to model data cleaning rules based on integrity constraints within the probabilistic framework of Markov logic (Section 3.1).

We describe how data repair leverages joint inference on probabilistic graphical models and allows us to treat data curation rules holistically, which obliterates the need to specify execution order (Section 3.2).

We present an extensive empirical study of holistic modeling and error prediction in data cleaning using Markov logic. Modelling data quality rules in a probabilistic way enables statistical inference of data quality errors. Our experimental evaluation reveals that the simultaneous treatment of data cleaning rules leads to higher accuracy in data curation. We show that joint inference for data correction prediction outperforms sequential execution of data cleaning rules on a dataset comprised of hundred thousands tuples with 10% of noisy values, and results in an improved F_1-score of 0.95 (Section 4).

2. CHALLENGES IN DATA CLEANING

Next, we showcase limitations of rule-based data cleaning. Therefore, we introduce the following data cleaning scenario:

Table I. CUSTOMER table (with errors)

id	firstname	lastname	street	city	zipcode	phone
c_1	Ron	Howard	1 Sun Dr.	Los Angeles	90001	12345
c_2	Max	Miller	12 Hay St.	Napa	94558	11234

Table II. TRANSACTION table (with errors)

id	item	firstname	lastname	street	city	zipcode	phone
t_1	iPhone6	R.	Howard	1 Sun Dr.	L.A.	null	null
t_2	Galaxy5	null	Miller	12 Hay St.	null	94558	11234
t_3	Nexus5	Howard	Ron	null	null	90001	12345

Consider two relational tables: the CUSTOMER table (c.f., Table I) records address and contact details of each customer. The TRANSACTION table (c.f., Table II) records each purchased item, together with the personal details entered by the customer during the purchase. The example data has several quality issues: (1) Missing values in the TRANSACTION table, indicated by *null* values; (2) incorrect values, e.g., the customer "Ron Howard" is involved in

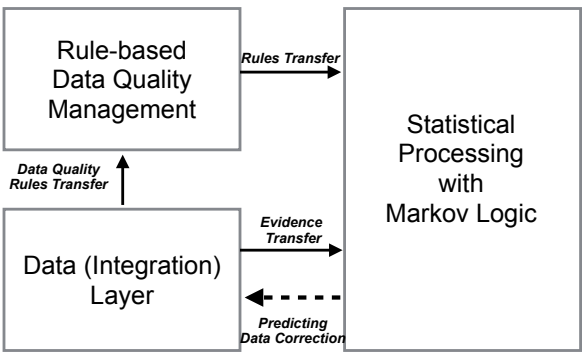


Fig. 1. Overview of the proposed data cleaning approach

transaction t_3 , but in the transaction table his name is falsely recorded as "Howard Ron"; (3) semantic heterogenities, which represent the same concept, e.g., the city of "Los Angeles" is sometimes entered into the table as "L.A." and the first name "Ron" is once abbreviated as "R.".

The example table is heavily corrupted, it is very hard to define data cleaning rules which do not introduce errors. Instead, we would prefer to define a number of *soft rules* that aggregate evidence.

For the *firstname* field in transaction t_1 in Table II, we could define a rule indicating that the values "R." and "Ron" refer to the same name, but cannot always be certain whether this is the case. To include additional evidence, we create the following matching dependency (MD) [Fan and Geerts 2012] to link transaction t_1 to customer c_1:

$$md : \text{TRANSACTION}[\textit{lastname}, \textit{city}, \textit{street}] = \text{CUSTOMER}[\textit{lastname}, \textit{city}, \textit{street}] \wedge \text{TRANSACTION}[\textit{firstname}] \approx \text{CUSTOMER}[\textit{firstname}] \rightarrow \text{TRANSACTION}[\textit{firstname}] \rightleftharpoons \text{CUSTOMER}[\textit{firstname}]$$

This rule (see operators explanation in Table III) shows how the notion of similarity may be included in rules, but only within first-order logic, i.e., the similarity condition is either *true* or *false*. In reality, we are able to determine a more fine graded similarity, i.e., some types of similarity that we deem to be more probable ("L.A." and "Los Angeles"), and others that are less probable ("R." and "Ron"). Moreover, we may have different levels of confidence regarding the functional dependencies or matching dependencies that we define.

The functional dependency [Abiteboul et al. 1995]: $fd : \text{TRANSACTION}([\textit{city}, \textit{phone}] \rightarrow [\textit{street}, \textit{zipcode}])$ declares that the two fields *city* and *phone* in the TRANSACTION table together uniquely determine the two fields *street* and *zipcode*. Although two instances of the customer "Ron Howard" in Table II are recorded, one is missing the *phone* value. In this case, the rule only applies when combined with additional data cleaning rules. The rule $cfd : \text{TRANSACTION}([\textit{zipcode}] \rightarrow [\textit{city}], T_1 = (90001 \mid\mid \text{Los Angeles}))$ denotes conditional functional dependency (CFD) [Fan and Geerts 2012] and states that every tuple in which the value for *zipcode* equals 90001 must have its *city* attribute set to "Los Angeles". In our example in Table II, this rule corrects the *null* value in transaction t_3.

3. PROPOSED APPROACH

We propose to model data cleaning rules in the *Markov logic* [Domingos and Lowd 2009] formalism and to leverage probabilistic inference for data cleaning. Markov logic is a knowledge representation system that combines first-order logic as a declarative language with probabilistic graphical models (undirected Markov Networks). We perform probabilistic inference on top of this representation. In the following sections, we illustrate how to transfer data cleaning rules into Markov logic and how to leverage probabilistic inference to determine data repair operations for a given dataset. Figure 1 shows a high-level overview of our approach. We distinguish between three major components: *I) Rule-based Data Quality Management*, where we specify data-agnostic quality rules to address different data quality issues;

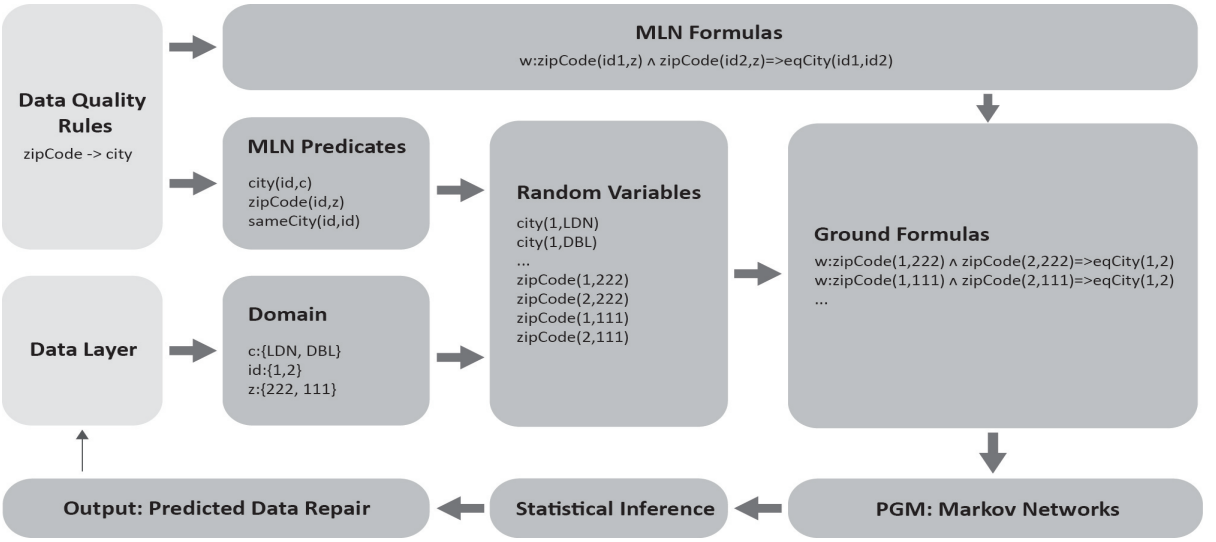


Fig. 2. In context of a data cleaning workflow, the Markov Logic Network grounding process consists of two phases: 1) MLN definition by (a) fixing MLN schema by defining observed and hidden predicates (b) domain, which is created from the existing data by considering the MLN schema, and (c) specification of weighted first-order logic formulas that represent data cleaning rules; 2) MLN instantiation by assigning truth values to all possible instantiations of the MLN predicates by consideration of the domain (Random Variables) and using these ground atoms in formulas. These ground formulas constitute a Markov Network in order to compute MAP inference and to estimate the most likely data repairs.

II) *Statistical Inference with Markov Logic* to compute data repairs based on probabilistic inference performed by the Markov logic framework; and the *III*) *Data Layer*, which interacts with the previous two components by providing both integrity constraints and evidence data from a number of data sources including relational and semi-structured data for Markov logic.

3.1 Compilation of Data Cleaning Rules to Predicate Calculus

The base of Markov logic programs is predicate calculus [Genesereth and Nilsson 1987], because Markov logic consists of first-order logic sentences. Therefore, we discuss a general method to compile formal constraint-based data cleaning rules into predicate calculus. We define data cleaning rules in the form of CFDs and MDs. For example, we express ϕ as *first-order logic* sentence, given the following functional dependency $\phi : X \rightarrow Y$ [Fagin 1982], [Burdick et al. 2015]:

$$\forall \mathsf{x}, \mathsf{y}_1, \mathsf{y}_2, \mathsf{z}_1, \mathsf{z}_2 \mathcal{R}(\mathsf{x}, \mathsf{y}_1, \mathsf{z}_1) \wedge \mathcal{R}(\mathsf{x}, \mathsf{y}_2, \mathsf{z}_2) \Rightarrow \mathsf{y}_1 = \mathsf{y}_2 \tag{1}$$

In following, we show that first-order logic sentences are crucial for the compilation of data cleaning rules into predictive models. Consider the logical equivalence of the data quality rule ϕ in 1 as a composite component, consisting of subcomponents such as atomic sentences (attribute), logical and quantified sentences (RHS and LHS in FD ϕ). We choose symbols that designate the elements of our conceptualization in order to describe the structure of ϕ in predicate calculus. In connection to the fundamentals of data quality management [Abiteboul et al. 1995], [Fan and Geerts 2012], we define the *vocabulary* to use for the compilation of data quality rules:

The *Universe of Disclosure* is specified as the set of all objects from domain $dom(U)$ that is fixed for the set of attributes $attr(\mathcal{R})$. A *term* names an object in the universe of discourse. We define *variables* to denote arguments in atoms and *constants* to denote data constants of a particular domain $dom(U_i)$ of the i-th attribute $U_i \in attr(\mathcal{R})$. To designate a tuple in relation $\mathcal{R}(x_1, x_2, \ldots, x_n)$, we use the atomic sentences *attr-*$X_1(\mathsf{id}, \mathsf{v}_1) \ldots$ *attr-*$X_\mathsf{n}(\mathsf{id}, \mathsf{v}_\mathsf{n})$ where *attr-*$X_\mathsf{i}(\mathsf{id}, \mathsf{v}_\mathsf{i})$ means that v_i is the attribute value of the i-th attribute in $\mathcal{R}(x_1, x_2, \ldots, x_n)$ of the id-th tuple in relation $\mathcal{R}$. *Relation constants* denote relations between several objects:

Similarity: *similar*$(\mathsf{x}_1, \mathsf{x}_2)$ means that x_1 similar to x_2 (e.g., by using different similarity measures like cosine or Jaccard similarity).

Equality: *equal-X*$(\mathsf{id}_1, \mathsf{id}_2)$ means that the values of the attribute X of two tuples id_1 and id_2 should be equal.

Matching: *match-X*$(\mathsf{id}_1, \mathsf{id}_2)$ means that values of two tuples id_1 and id_2 of the attribute X are identified to match.

Custom Predicate: encodes diverse semantic constraints (e.g. *contains*, *between*, *less* etc.), which are not part of the data constraints.

Given this vocabulary, we describe our conceptualization of the data quality rules with predicate-calculus sentences. For example, we compile the functional dependency $\phi : X \rightarrow Y$ and the first-order logic sentence in Formula 1 to the corresponding Markov logic formula as follows:

attr-X$(\mathsf{id}_1, \mathsf{x}) \wedge$ *attr-X*$(\mathsf{id}_2, \mathsf{x}) \Rightarrow$ *equal-Y*$(\mathsf{id}_1, \mathsf{id}_2)$

Analogously, we translate a matching dependency $\mu : \mathcal{S}_1[x_1] \approx \mathcal{S}_2[x_2] \rightarrow \mathcal{S}_1[y_1] \rightleftharpoons \mathcal{S}_2[y_2]$ on a database instance $\mathcal{D}$ with two relational schemas $\mathcal{S}_1$ and $\mathcal{S}_2$ as follows:

attr-X/S1$(\mathsf{id}_1, \mathsf{x}_1) \wedge$ *attr-X/S2*$(\mathsf{id}_2, \mathsf{x}_2) \wedge$ *similar*$(\mathsf{x}_1, \mathsf{x}_2) \Rightarrow$ *S1/match-Y/S2*$(\mathsf{id}_1, \mathsf{id}_2)$

If we specify a matching dependency using two relations, we need to mark the attributes corresponding to a relation with that relation *attr-X/*$\mathcal{R}$. Furthermore, as introduced above, we use predicates to represent operators as shown in Table III.

Table III. Markov logic predicates used for data cleaning.

Concept	Operator	Predicate
Similarity	$\mathcal{S}_1[x_1] \approx \mathcal{S}_2[x_2]$	*similar*$(\mathsf{x}_1, \mathsf{x}_2)$
Equality	$y_1 = y_2$	*equal-Y*$(\mathsf{id}_1, \mathsf{id}_2)$
Matching	$\mathcal{S}_1[y_1] \rightleftharpoons \mathcal{S}_2[y_2]$	*S1/match-Y/S2*$(\mathsf{id}_1, \mathsf{id}_2)$

We assume that integrity constraints have already been determined by methods reviewed in [Liu et al. 2012], or have been specified by domain experts manually. To demonstrate the compilation of the data cleaning rules into Markov logic, have a look at the CFD from the motivation example in Section 2:

$$\mathsf{fd} : \text{TRANSACTION}([\textit{city}, \textit{phone}] \rightarrow [\textit{street}, \textit{zipcode}])$$

To enable straightforward compilation, we assume that CFDs are provided in normal form. This means if $\psi(X \rightarrow Y_1, Y_2, \ldots, T_p)$, then ψ will be decomposed into several CFDs where $RHS(\psi)$ (right hand side of ψ) becomes a single attribute: $\psi_1(X \rightarrow Y_1, T_p)$, $\psi_2(X \rightarrow Y_2, T_p) \ldots$. Following the normalization rule for functional dependencies, we split the fd rule into two rules:

$\mathsf{cfd}_1 : \text{TRANSACTION}([\textit{city}, \textit{phone}] \rightarrow [\textit{street}], \mathsf{t}_1 = (\_, \_ \parallel \_))$

$\mathsf{cfd}_2 : \text{TRANSACTION}([\textit{city}, \textit{phone}] \rightarrow [\textit{zipcode}], \mathsf{t}_2 = (\_, \_ \parallel \_))$

In accordance with [Fagin 1982], we represent cfd_1 and cfd_2 as two first-order logic formulas:

1) $\forall$ *city*, *phone*, *street*$_1$, *street*$_2$TRANSACTION(*city*, *phone*, *street*$_1$) $\wedge$TRANSACTION(*city*, *phone*, *street*$_2$)$\Rightarrow$ *street*$_1$ = *street*$_2$

2) $\forall$ *city*, *phone*, *zip*$_1$, *zip*$_2$ TRANSACTION(*city*, *phone*, *zip*$_1$)$\wedge$TRANSACTION(*city*, *phone*, *zip*$_2$)$\Rightarrow$ *zip*$_1$ = *zip*$_2$

Once we have formulated our integrity constraints as first-order logic formulas, we translate them into Markov logic syntax. Given that every attribute from the schema TRANSACTION is expressed as a predicate, we need two predicates, namely *city*$(\mathsf{id}, \mathsf{city})$ and *phone*$(\mathsf{id}, \mathsf{phone})$ to encode the LHS of fd. They indicate the values for the fields *city* and *phone* for each tuple (Table IV gives the full example of predicate translation and data translation from

Table IV. MLN declaration process and creation of grounded atoms for Tuple 2 in the TRANSACTIONS example table.

Phase	Example
1) Schema definition	t_2(item, firstname, lastname, street, city, zipcode, phone)
2) Observed predicates MLN declaration	*firstname*(id, firstname) *lastname*(id, lastname) *street*(id, street) *city*(id, city) *zip*(id, code) *phone*(id, num)
3) Data	t_2(Galaxy 5, NULL, Miller, 12 Hay St., NULL, 818, 11234)
4) Grounded (evidence) atoms	*item*(2, Galaxy5) *lastname*(2, Miller) *street*(2, 12HaySt.) *zip*(2, 818) *phone*(2, 11234)

into grounded atoms). Additionally, we define two predicates for our data quality rule, namely *equal-street*(id, id) and *equal-zip*(id, id). These predicates model equality of two values of the attribute *street* respectively *zip* (as denoted by the fd rule above):

$1)$*city*$(\mathsf{id}_1, \mathsf{city}) \wedge$ *city*$(\mathsf{id}_2, \mathsf{city}) \wedge$ *phone*$(\mathsf{id}_1, \mathsf{phone}) \wedge$*phone*$(\mathsf{id}_2, \mathsf{phone}) \Rightarrow$ *equal-street*$(\mathsf{id}_1, \mathsf{id}_2)$

$2)$*city*$(\mathsf{id}_1, \mathsf{city}) \wedge$ *city*$(\mathsf{id}_2, \mathsf{city}) \wedge$ *phone*$(\mathsf{id}_1, \mathsf{phone}) \wedge$*phone*$(\mathsf{id}_2, \mathsf{phone}) \Rightarrow$ *equal-zip*$(\mathsf{id}_1, \mathsf{id}_2)$

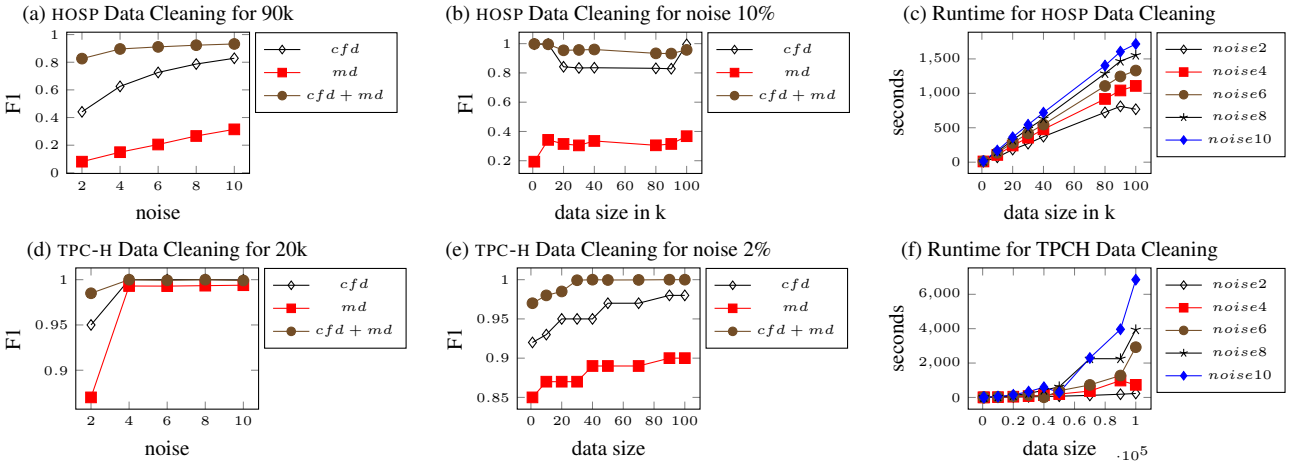
Note that using the same arguments in predicates *city*(id, city) and *phone*(id, phone) encodes equality of the corresponding values into the Markov logic program. The two different tuples are distinguished by id_1 and id_2. These data quality rules form the basis for a Markov logic program. A particular advantage of Markov logic is its *modularity* while modeling. We consider each declared data quality rule as an "atomic" probabilistic model. In combination, these small models form a "compound" model. Having declared the data quality rules by capturing all correlations and constraints, we now infer potential predicates, such as *equal-street*(id, id). This predicate holds the information about possible repairs on the attribute *street* in the TRANSACTIONS table. Reasoning about such hidden predicates allows us to decide which attributes get a particular repair.

3.2 Data Repair as Joint Inference

Markov logic defines a knowledge representation system that combines first-order logic as a declarative language with probabilistic graphical models (undirected Markov Networks MLNs) on top of which we perform probabilistic inference. Semantically, a MLN is a log-linear model, which defines the probability distribution over possible worlds, in our case all possible repairs in the database.

We create MLNs by writing a set of first-order logic rules (c.f., Section 3.1) with weights by using predicates that represent relations between these objects. In order to specify *soft* and *hard* rules in our experiments we set the weight of *soft* rules to 1.0, whereas *hard* rules are assigned infinite weights. We distinguish between observed and hidden predicates. *Observed predicates* are relations between objects which exist in a given dataset: *attr-*$X_1(\mathsf{id}, \mathsf{v}_1)$ ... *attr-*$X_\mathsf{n}(\mathsf{id}, \mathsf{v}_\mathsf{n})$. The predicate *attr-*$X_\mathsf{i}$ has value v_i on tuple id. In addition, an MLN may have a number of *hidden predicates*, which are not present in the input data, but may be inferred through rules. In our case, *equal-*$Y(\mathsf{id}_1, \mathsf{id}_2)$ and *S1/match-Y/S2*$(\mathsf{id}_1, \mathsf{id}_2)$ are hidden. We define the MLN in such a way that reasoning about hidden predicates given evidence and data cleaning rules allows us to determine data repair operations. In other words, we perform an inference task as prediction for data cleaning. An important step in our method is the *grounding* of the MLN (illustrated

Table V. Evaluation of the data repair method based on Markov logic applied on the HOSP and TPC-H datasets. (a)-(c) Data repair on HOSP with an extended Markov logic method. (d)-(f) Experimental study of the Markov logic data cleaning on synthetic dataset TPC-H.



in Figure 2 in details). We take the content of the database (a set of tuples) and produce a set of *grounded* predicates by replacing predicate variables with domain constants. We use these groundings in the joint inference for data cleaning.

Given an MLN that models data cleaning rules, let $q \in \mathcal{L}^n$ denote a hidden predicate with n *literals* (random variables) $L_1, ..., L_n$, where each literal L_i has 2 discrete states, $L_i = \{0, 1\}$. Then, the MLN is a joint distribution on $L_1, ..., L_n$ that is specified by a vector $\phi(q)$ of d integer values, where each element represents the number of true groundings of the corresponding literal in the formula and d denotes the maximum number of literals in a given formula. Additionally, we have a weight/parameter vector $\theta \in \mathcal{R}^d$:

$$Pr\left(q|\theta\right) = \frac{1}{Z(\theta)} \exp\left(\langle\theta, \phi(q)\rangle\right), Z(\theta) = \sum_{q \in \mathcal{L}^n} \exp\left(\langle\theta, \phi(q)\rangle\right)$$

where $\langle\theta, \phi(q)\rangle$ denotes a dot product. $Z(\theta)$ is the normalization constant also called the *partition function*. Since the partition function is a constant and the exponential is monotonic, finding the MAP assignment in our data cleaning problem is equivalent to finding the assignment q_M that maximizes the probability $Pr\left(q|\theta\right)$.

The output of the inference are data repair operations; e.g., the hidden predicate *equal-street* may be determined to have (among other groundings) the following likely value: *equal-street*$(1, 3)$, indicating that the *street* field for transaction 3 should have the same value as the *street* field of transaction 1. In this case, the data repair operation is to replace the NULL value in transaction 3 with the address "1 Sun Dr.". The inference produces the most likely state of the entire Markov Logic Network with regards to all integrity constraints. The probabilities for the hidden predicates are therefore influenced by *all* defined data quality rules. By running the inference over the entire database, we predict the most likely data repairs for our dataset by determining the most likely grounding of the hidden predicates and this reduces to computing MAP inference on the MLN model.

The difficulty in designing algorithms for MAP inference arises when finding an efficient way to reason about the large number of groundings. Amongst the numerous techniques that are available to solve MAP inference problems, we choose to cast the inference problem as an *integer linear program* (ILP)[Sontag 2010]. Another competitive approach called *message passing* performs *belief propagation* along the edges of the graphical model. Although, message passing is straightforward to implement, it has troubles converging [Schwing et al. 2011], [Felzenszwalb and Huttenlocher 2006] and tends not to give as good results as ILP [Noessner et al. 2013].

Table VI. Scalability of the data cleaning for TPC-H and HOSP datasets (with fixed noise 4%).

dataset	numer of tuples	number of formulas	runtime (sec)
HOSP	63K	21	242
	83K	21	477
	143K	21	1107
TPC-H	20K	15	41
	40K	15	131
	100K	15	732

Table VII. Markov logic predicates used in data quality rules.

	HOSP	TPC-H	MSAG
observed predicates	*providerNr*/HOSP(hid, pn) *hospitalName*/HOSP(hid, n) *address*/HOSP(hid, add) *city*/HOSP(hid, c) *state*/HOSP(hid, st) *zipCode*/HOSP(hid, code) *phoneNumber*/HOSP(hid, numb) *condition*/HOSP(hid, cond) *measureCode*/HOSP(hid, mcode) *score*/HOSP(hid, score) *zip*/ZIPCODE(zid, code) *state*/ZIPCODE(zid, st)	*custKey*(id, key) *name*(id, n) *addr*(id, add) *natKey*(id, nkey) *phone*(id, ph) *acc*(id, a) *mrkt*(id, m) *orderKey*(id, okey) *totalPrice*(id, p) *orderDate*(id, d) *orderPriority*(id, pr) *clerk*(id, c)	*publishYear*(paperid, pubyear) *author*(paperid, authorid) *affiliation*(paperid, affilid) *inRange*(pubyear, pubyear) *originAffiliationName*(affilid, oname) *normalAffiliationName*(affilid, nname)
hidden predicates	*equal-HospitalName*/HOSP(hid, hid) *equal-Address*/HOSP(hid, hid) *equal-City*/HOSP(hid, hid) *equal-State*/HOSP(hid, hid) *equal-ZipCode*/HOSP(hid, hid) *equal-PhoneNumber*/HOSP(hid, hid) *equal-Condition*/HOSP(hid, hid) HOSP/*match-State*/ZIPCODE(hid, zid) HOSP/*match-ZipCode*/ZIPCODE(hid, code)	*equal-Names*(id, id) *equal-Addr*(id, id) *equal-Natkey*(id, id) *equal-Phone*(id, id) *equal-Acc*(id, id) *equal-Mrkt*(id, id) *match-Phone*(id, id) *match-Addr*(id, id)	*equal-Affiliation*(paperid, paperid) *equal-OriginNames*(oname, oname) *equal-OriginNamesByPaperId*(paperid, paperid) *equal-NormalNames*(nname, nname) *equal-NormalNamesByPaperId*(paperid, paperid) *missingOriginName*(paperid, oname)

Table VIII. F_1 measure comparison of the jointly modeled data cleaning rules based on CFD and MD to the baseline system [Dallachiesa et al. 2013]. The experiments conducted on fixed HOSP data size $90K$ and different noise level from 2% to 10%.

System	2%	4%	6%	8%	10%
Baseline system [Dallachiesa et al. 2013]	0.69	0.71	0.75	0.83	0.85
SLR based approach	0.82	0.89	0.90	0.91	0.93

4. EXPERIMENTAL STUDY

We evaluate our method through an experimental study on well-known datasets, which have been used for assessing other data cleaning systems [Dallachiesa et al. 2013], [Chu et al. 2013], [Geerts et al. 2013], [Bohannon et al. 2005].

4.1 Experimental Setting

We conduct our experiments on the following real-life and synthetic datasets.

HOSP. The HOSP dataset has been published by the US Department of Health & Human Services[1]. This dataset comprises 9 attributes: *addr*, *city*, *cond*, *country*, *hospname*, *measure*, *phone*, *state*, *zip*. We use 6 CDFs and one MD, which have been manually designed. These data quality rules have been generously provided to us by the researchers Dallachiesa et al. from [Dallachiesa et al. 2013]. They define an MD that makes use of another table, namely US ZIP codes: ZIPCode[2]. This additional dataset contains $43K$ tuples with two attributes: *zip* and *state*. The MD defines that if two tuples from *hosp* and ZIPCode possess the same zip code values, and the state values are distinct, then the state value from the ZIPCode table should be adopted.

TPC-H. The TPC-H[3] dataset is well-known dataset used in decision support benchmarks for databases. For our experiments we use two relations, *Customer* and *Orders*, which we join in order to introduce duplications on the *Customer* relations data. The resulting dataset consists of 17 attributes of schema T: *c_custkey*, *c_name*, *c_address*, *c_nationkey*, *c_phone*, *c_acctbal*, *c_mtksegment*, *c_comment*, *o_orderkey*, *o_custkey*, *o_orderstatus*, *o_totalprice*, *o_orderdate*, *o_orderpriority*, *o_clerk*, *o_shippriority*, *o_comment*.

Dirty Data. We introduce noise into the relational datasets HOSP and TPC-H to produce dirty data. Our methods handles several kinds of noise: missing values, errors from the active domain, and typos. We consider the initial data to be clean and therefore use it as ground truth. Additionally, we manually assess that the ground truth datasets are consistent with respect to the CFDs and MDs. Afterwards, we insert noise into the datasets. We conduct our experiments on the two datasets with different noise rates ranging from noi%=2% to noi%=10%. We introduce this noise into different dataset sizes ranging from one thousand to one hundred thousand data points. The noise rate depicts the ratio of the number of erroneous values to the total number of values in the dataset. We only introduce noise to the attributes which are involved in data quality rules.

MICROSOFT ACADEMIC GRAPH (MSAG). Data quality issues are massively present in the context of the web due to the integration of heterogeneous data from different sources. Thus we include a third dataset - MICROSOFT ACADEMIC GRAPH (MSAG)[4] [Sinha et al. 2015] - to assess our method on web data cleaning. MSAG is a heterogeneous entity graph comprised of six types of entities that model real-life academic relationships: field of study, author, institution, paper, venue, and conference instances. The raw data has been obtained from different sources (academic publishers and web-pages indexed by Bing search engine) and is organized in the form of a connected graph schema. For our experiments, we select three entities from the whole graph: *author*, *organization* and *paper* entities. This part of MSAG is of interest because it reveals important characteristics of the extracted web data, such as missing and inconsistent values. In particular, we discover that there are inconsistencies in organization names for the same author. Furthermore, a number of entries suffer from missing affiliation by an author. In order to model and run data cleaning on this web dataset, we consider the attributes of the selected entities in MSAG, as presented in Table IX.

Table IX. Entities ***Paper***, ***Author*** and ***Organization*** and their attributes that have been used in experiments for Web data cleaning with Markov logic.

Paper	***Author***	***Organization***
paper_id	*author_id*	*affiliation_id*
publish_year		*origin_name*
		normalized_name

[1] http://www.medicare.gov/hospitalcompare/Data/Data-Download.html
[2] http://databases.about.com/od/access/a/zipcodedatabase.htm
[3] http://www.tpc.org/tpch/
[4] http://research.microsoft.com/en-us/projects/mag/

Data Quality Issues. We reproduce data quality issues that we observe in MSAG: namely missing values. In order to a create gold standard to assess our data cleaning method, we proceed in the following way: We remove one affiliation entry from each author if there are more than three publications made by the same affiliation. Thus, we create a dataset with missing *affiliation_id*, *origin_name*, and *normalized_name* attributes. We thereby obtain graph data where 37% of all authors have one missing edge to theirs affiliation; 27% are missing two edges for theirs institutions; 15% miss three edges. Almost 21% of authors suffer from missing more than three edges.

Evaluation metrics. To assess the accuracy of the data cleaning framework on relational data, we use *Precision (P)*, *Recall (R)*, and *F-measure (F_1)*. We acquire master data (referred to as *gold standard*), which is clean and correct. We assess the efficiency of our method by running experiments on datasets of different sizes ranging from one thousand to one hundred thousand tuples each. We leverage a state-of-the-art inference engine for Markov logic called *RockIt* [Noessner et al. 2013] and the Gurobi solver version 5.6.3. All our experiments apply MAP inference for statistical relational learning. We execute the experiments on a Linux server with an Intel 3.4GHz 4 Cores CPU and 16 GB of RAM.

4.2 Holistic Data Cleaning: Deduplication and Accuracy

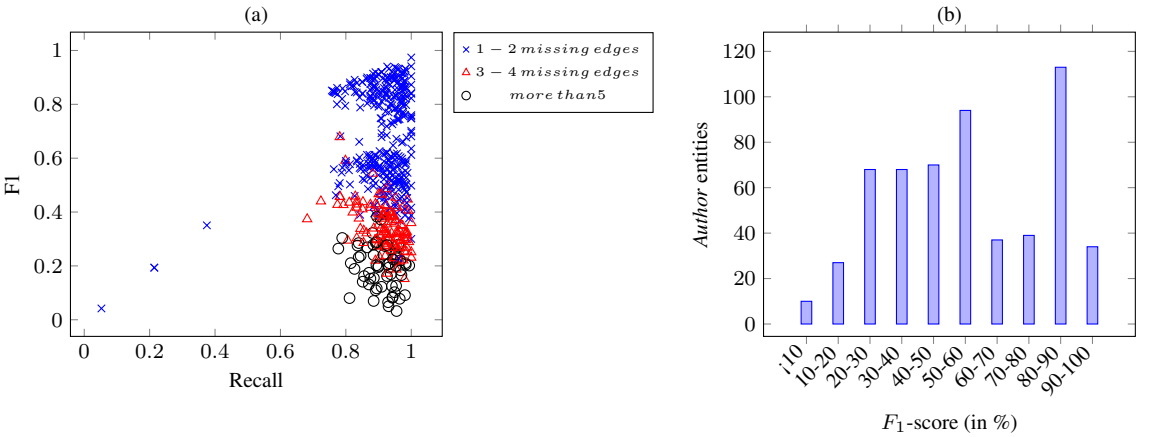
In this experiment, we show that capturing the data issue interaction increases the overall accuracy in data cleaning. In particular, we study the connection between deduplication and improved data accuracy. We evaluate the accuracy of our method on different noise rates and different dataset sizes. We introduce noise by adding either typos or replacing values of attributes (active domain errors). We do not distinguish between these two kinds of errors. Table V shows the results of using Markov logic programs for data cleaning. The plots (a)-(c) illustrate the results for HOSP, while plots (d)-(f) depict the results for TPC-H.

In the first series of experiments, we fix the size for HOSP and TPC-H to 90K respectively 20K tuples, while varying the noise rate from 2% to 10%. The x-axis in Table V (a) and (d) represents the noise rate, the y-axis shows the corresponding F_1 measure values. First, we compare the results of separate executions of CDFs and MDs. After that we re-run the experiments with combined CFDs and MDs. When CFDs and MDs are modeled together in a single model for identifying dirty data, they are treated jointly and hence our method automatically picks the order of the data quality rules execution. The provided results in Table V (a) and (d) clearly demonstrate that jointly modeling CDFs and MDs improves the overall result because joint execution involves two processes simultaneously: matching (for values deduplication) and repair (for erroneous values). For this reason repairing supports matching, and by identifying matches we are able to fix erroneous values.

In particular, the low F_1 score of MDs in Table V (a)-(b) and (d)-(e) results from low precision and high recall. However, By adding CFDs to MDs the overall F_1 score improves. The counterintuitive upward trend of the F_1 values while increasing noise in Table V (a) and (d) results from the *cutting plane inference* (CPI) [Riedel 2008] behavior. In all experiments, we use the CPI algorithm, which leverages optimization techniques such as integer linear programming (ILP) [Riedel 2008], [Noessner et al. 2013] by maximizing the objective function under set of constraints (every formula in MLN is being converted into an ILP constraint). CPI performs exact MAP inference and is guaranteed to converge in a finite number of steps [Riedel 2008]. For almost perfect data with less noise, the algorithm requires only a short runtime and converges fast by finding the approximately maximum objective score. This however results in detecting less data violations. Looking at the runtime in Table V (c) and (f), we recognize that, with increasing noise, the underlying solver calculates a solution until the maximum number of iterations is reached. For that reason, more data quality rule violations are encountered.

In the second series of experiments, we fix the overall noise rate to 10% on HOSP (c.f., Table V (b)) and to 2% on the synthetic TPC-H (c.f., Table V (e)). We run three combinations of data cleaning rules on dataset sizes ranging from one thousand to one hundred thousand tuples. In Table V (b) respective (e), the x-axis represents the data size and the y-axis shows the F_1-measure values. These two plots in (b) and (e) tell us that due to convergence guarantees of CPI our data cleaning method delivers robust results independently from the data size despite of the increasing runtime of the algorithm (c.f., Table V (c) and (f)) by growing the size of the dataset.

Table X. (a) MSAG cleaning: Recall and F1 values; (b) MSAG Experiments



Our results (c.f., Table VIII) are in line with the observations made in [Dallachiesa et al. 2013]. There, the NADEEF system demonstrates an overall F_1-measure improvement. With a holistic treatment of FDs and MDs, NADEEF achieves an F_1-measure of 0.85. We compare our results to theirs, as our results are based on the identical data cleaning problem, the same evaluation methodology, as well as the same dataset HOSP. Although their system demonstrates better performance for MD rules than ours, we get higher F_1-score values for the joint execution of the deduplication and accuracy rules: We achieve an overall performance of 0.93 and 0.99 for HOSP respectively TPC-H datasets. Therefore, we report an accuracy improvement by a factor of 1.2 against this reference system.

We conclude from this experiment that the joint modeling and joint inference improves the accuracy achieved by running single data quality rules. We also study the runtime of our method for different data sizes and noise rates. Table V (c) for HOSP and Table V (f) for TPC-H show the general trend for the runtime values (y-axis) for every data size (x-axis) setting. Each plot denotes different noise percentages. We see that the runtime on both, real-life and synthetic data, shows an upward trend. Cleaning datasets with joint inference will take longer the larger the data is and the more noise it contains. In particular, for the real-life HOSP data of size $100K$ with increasing noise rate, we observe that the runtime increases by a factor 1.2 for every noise setting.

Additionally, Table VI provides detailed runtime values for different data sizes of HOSP and TPC-H by fixing the noise to 4%. We observe that while we increase the data size from 20 to 143 thousands tuples, the average runtime growth rate is 1.9. This is again a characteristic of the underlying MAP inference algorithm [Riedel 2008]: Having less noise causes the algorithm converge faster, hence results in faster run time values for data with less noise.

4.3 Holistic Data Cleaning: Missing Value and Consistency Issues Interaction

We extend the applicability of Markov logic to web data and provide initial results on cleaning the highly imperfect graph structured MSAG dataset. We study the efficiency of data cleaning on web data by leveraging the connection between information completeness and data consistency. These two data issues interact with each other: missing values imputation helps to fix inconsistencies, and by correcting values, we identify missing entities.

In this experiment, we partition our data and run inference for each author in isolation. We obtain a subgraph per *Author*-entity with at least 10 *Paper-Author* edges. Next, we randomly select 600 *Author*-entities.

Table X (a) shows the accuracy of our approach on the sample of MSAG dataset. We show the accuracy as relation between recall (x-axis) and F_1-score (y-axis). We distinguish between three kinds of *Author*-nodes (which are marked separately in Table X (a)): nodes with one or two missing *Author-Organization* edges, nodes with three or four missing values for the *Author Organization* connection, and finally nodes with more than 5 missing edges. Additionally, Table X (b) provides another perspective on this experiment by revealing the exact distribution of corrected *Author*-

entities (x-axis) by F_1-score (y-axis). The result shows the following: The method demonstrates overall F_1 score greater than 50% and recall greater than 78% for *Author*-entities with one to two missing edges. Starting from three missing edges, we still achieve a recall ranging from 0.8 to 1.0, though the precision (and therefore F_1 score) drops. This happens because our approach selects more false positives with an increasing number of missing values. This experiment tells us that our method produces satisfying results on web data with very little noise only (e.g., missing values). In future work, extending data cleaning rules with domain specific information should be studied further in order to reduce the false positives rate. We are not able to compare ourselves to another system directly here, because the MSAG dataset has only recently been published and related work in data cleaning systems only provides results for cleaning relational data.

4.4 Impact of Rule Execution Order

Next, we study the effects of different orders of data cleaning rule execution, to show that specifying the optimal order of rules manually is hardly achievable [Dallachiesa et al. 2013]. We investigate whether it is beneficial to leverage joint inference for simultaneous rule execution instead of manual specification of the optimal order of data cleaning rule execution. We verify the result in Figure 3 by investigating the F_1-score (y-axis) distribution of various data quality settings executed on varying noise rates (x-axis). We run our Markov logic program on the attributes *state* and *zip* of the HOSP dataset because they both participate in CFD and MD. Furthermore, we fix the dataset size at 90,000 tuples with a noise rate varying from 2% to 10%. Each experiment consists of three parts: First, we run the MD rules and then the CFDs, which gives us the overall worst accuracy. This results agree with the previous experiment about the joint modeling data cleaning rules (c.f., Table V (a) and Section 4.2). The MD rules perform poorer than CFDs. The overall F_1 score ranges between 0.01 and 0.02, which we explain by low precision and recall values due to error propagation from MDs to CFDs. In the second part of the experiment, we change the sequence of execution to running CFD rules before MD rules. This slightly improves the F_1 scores by increasing them from 0.1 to 0.3 for different noise values. We attribute this to the fact that CFDs initially detect more violations. However, analogous to the previous part, error propagation leads to unsatisfactory results.

In the third and final part of the experiment, we perform the simultaneous execution of CFD and MD rules, where we model matching (for values deduplication) and repair (for erroneous values) processes together. Here we see a rapid increase of all accuracy values from 0.86 for 2% noise to 0.95 on 10% noise in comparison to the previously performed sequential execution of data cleaning rules. The growing trend of the F_1 scores has the same explanation as the results of the experiment about holistic data cleaning (c.f., Table V (a) and Section 4.2): namely the functionality of the CPI algorithm. These results confirm our hypotheses that it is highly beneficial to execute multiple data cleaning rules simultaneously.

The NADEEF system [Dallachiesa et al. 2013], which also treats various types of rules holistically, ran an analogous experiment. Note that, on data with 10% noise, we improve the F_1-score by 10% over the values reported by NADEEF.

4.5 Modeling Data Cleaning Rules

To assess the usability of our method, we adopt the research methodology from UMUX-LITE [Lewis et al. 2013] and discuss two items from the UMUX-LITE questionnaire: "Markov logic capabilities meet the requirements of data cleaning systems" and "Markov logic is easy to use". As explained in the previous Sections 3, 4.2, 4.3 and 4.4, Markov logic meets the main requirements of data cleaning systems such as *holistic* data quality rules treatment [Fan et al. 2014], [Dallachiesa et al. 2013], automation [Stonebraker et al. 2013] and heterogeneous rules incorporation [Chu et al. 2013]. In the following, we focus on the second item from UMUX-LITE - "Markov logic is easy to use" - and provide our experience in how we modeled data cleaning rules by using Markov logic on all three datasets HOSP, TPC-H and MSAG.

HOSP Quality Rules. In our data cleaning method for the HOSP data, we use 6 manually designed CDFs and one MD, which result in 15 normalized CFDs. One MD rule is transformed into 2 formulas. All data cleaning rules are positive. Finally, all interleaved rules are translated into 21 Markov logic formulas. In Table XI, we provide an example of

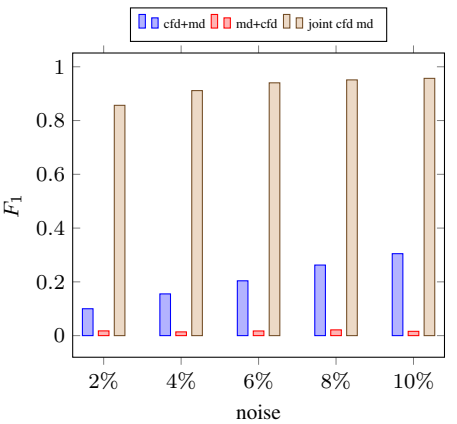


Fig. 3. The evaluation of the different experimental settings of the execution order of data cleaning rules translated into Markov Logic.

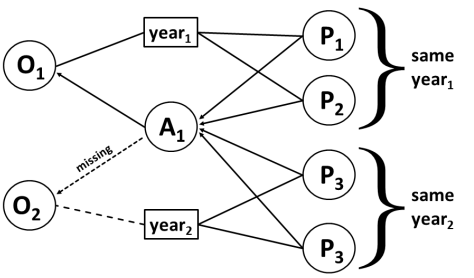


Fig. 4. The part of the MSAG dataset illustrating missing organization values. Markov logic allows us to capture the following evidence: if two papers of the same author have been published in the same year, then they may be published by the author of the same organization. Nodes notation: A - denotes *Author* entity; O - *Organization* and P - *Paper* entities. Missing edges are marked as dashed lines.

these data quality rules, which are defined on pairs of tuples. The MD rule is specified on two relations. Markov logic predicates used for data quality formulas are shown in Table VII. After transforming the 100k HOSP tuples into Markov logic grounded atoms, the resulting data comprises 1.3M evidence atoms, which are used for inference. We empirically observe that extending the data quality rules set for additional conditions makes consideration of similar tuples unnecessary and reduces the search space and therefore converges much faster than the *"pure"* model. This means that each first order logic formula, which represents the RHS of the normalized data quality rule $attr(\mathsf{id1}, \mathsf{v1}) \wedge attr(\mathsf{id2}, \mathsf{v2})$, becomes an inverse part $!attr(\mathsf{id1}, \mathsf{v2}) \wedge !attr(\mathsf{id2}, \mathsf{v1})$. This additional part denotes that we consider only tuples with different values. Here the normalized CFD rules are being compiled as demonstrated in the Table XI.

TPC-H Quality Rules. We write 9 CFDs and 3 MDs for this dataset. One example of the rules is a CFD that states if two tuples match on *c_custkey*, then they should match on the *c_name* and *c_address* attributes. MDs are designed on the same schema TPC-H *(T, T)*. These MDs state that if the LHS is similar for any pair of tuples (t_1, t_2), then the attribute values on the RHS should be identified. We provide an except of the data quality rules we created for TPC-H in Table XI. Table VII shows the Markov logic predicates which we use for data quality rules. After transforming the 100k TPC-H tuples into Markov logic grounded atoms, the resulting data comprises 1 Mio evidence atoms, which are leveraged by the inference.

MSAG Quality Rules. Due to the graph nature of the MSAG data (c.f., Figure 4), we develop data quality rules, which are based on CFDs, *extended* CFDs [Chen et al. 2009], and equality axioms, such as *symmetry* and *transitivity*. For the *Paper-Author-Organization* subgraph we define two CFDs, one extended CFD and 8 additional rules that comprises equality axioms for hidden predicates. Considering the semantical meaning of the data, we profit from the ability to add supporting knowledge into the data cleaning process. For example, the CFD $\mathrm{M}([author\_id, year] \rightarrow \rightarrow [affiliation\_id], \mathsf{t1} = (\_, \_ \parallel \_))$ allows us to capture all missing affiliations by the same author, which published in the same year. In real life, we know that in academia an average contract lasts around 2-3 years. Therefore we will extend our search range by incorporating this knowledge in the form of a predicate $inRange(\mathsf{year}, \mathsf{year})$. Additionally we enable capturing more missing values by adding equality axioms. For example, the hard rule $equal\text{-}Affiliation(\mathsf{id}_1, \mathsf{id}_2) \wedge equal\text{-}Affiliation\ (\mathsf{id}_2, \mathsf{id}_3) \Rightarrow equal\text{-}Affiliation(\mathsf{id}_1, \mathsf{id}_3)$ denotes a transitive relationship

Table XI. Modeling data cleaning rules as Markov logic programs (an excerpt). To specify *soft* rules, the weights w_i are set to 1.0 and *hard* rules are marked as rules with infinite weights: ∞.

Dataset	*Data Cleaning Rules*	*Markov Logic Formulae*
HOSP	cfd_1 : HOSP([*zip*] → [*state, city*], t1 = (_ ‖ _, _))	w_1 : *zip*/HOSP(id1, code) ∧ *zip*/HOSP(id2, code) ∧ *state*/HOSP(id1, s1) ∧ *state*/HOSP(id2, s2) ∧ !*state*/HOSP(id1, s2) ∧ !*state*/HOSP(id2, s1) ⇒ *equal-state*/HOSP(id1, id2)
		w_2 : *zip*/HOSP(id1, code) ∧ *zip*/HOSP(id2, code) ∧ *city*/HOSP(id1, c1) ∧ *city*/HOSP(id2, c2) ∧ !*city*/HOSP(id1, c2) ∧ !*city*/HOSP(id2, c1) ⇒ *equal-city*/HOSP(id1, id2)
	md_1 : HOSP[*zip*] = ZIPCODE[*zip*] ∧ HOSP[*state*] ≠ ZIPCODE[*state*] → HOSP[*state*] ⇌ ZIPCODE[*state*]	w_3 : *zip*/HOSP(id1, code) ∧ *zip*/ZIPCODE(id2, code) ∧ *state*/HOSP(id1, s1) ∧ *state*/ZIPCODE(id2, s2) ⇒ HOSP/*match-State*/ZIPCODE(id1, id2)
TPC-H	cfd_1 : T([*c_custkey*] → [*c_name,c_address*], t1 = (_ ‖ _, _))	w_1 : *custKey*(id1, key) ∧ *custKey*(id2, key)∧ *name*(id1, n1) ∧ *name*(id2, n2) ∧ !*name*(id1, n2) ∧ !*name*(id2, n1) ⇒ *equal-Names*(id1, id2)
		w_2 : *custKey*(id1, key) ∧ *custKey*(id2, key)∧ *addr*(id1, addr1) ∧ *addr*(id2, addr2) ∧ !*addr*(id1, addr2) ∧ !*addr*(id2, addr1) ⇒ *equal-Names*(id1, id2)
	md_1 : T[*c_address*] = T[*c_address*] → T[*c_phone*] ⇌ T[*c_phone*]	w_3 : *addr*(id1, addr) ∧ *addr*(id2, addr) ∧ *phone*(id1, phone1) ∧ *phone*(id2, phone2) ∧ ⇒ *match-Phone*(id1, id2)
MSAG	cfd_1 : M([*author_id, year*] → [*affiliation_id*], t1 = (_, _ ‖ _))	w_1 : *author*(pid1, aid1) ∧ *author*(pid2, aid2)∧ *publishYear*(pid1, y) ∧ *publishYear*(pid2, y) ⇒ *equal-Affiliation*(pid1, pid2)
	eCfd_1 : M([*author_id, year*] → [*affiliation_id*], t1 = (_, diff(_) ≤ 2 ‖ _))	w_2 : *author*(pid1, aid1) ∧ *author*(pid2, aid2)∧ *publishYear*(pid1, y1) ∧ *publishYear*(pid2, y2)∧ *inRange*(y1, y2) ⇒ *equal-Affiliation*(pid1, pid2)
		$symmetry$: ∞ : *equal-Affiliation*(pid1, pid2) ⇒ *equal-Affiliation*(pid2, pid1)
		$transitivity$: ∞ : *equal-Affiliation*(pid1, pid2) ∧ *equal-Affiliation*(pid2, pid3) ⇒ *equal-Affiliation*(pid1, pid3)

between three different entities *Organization*. In total, our Markov logic program then consists of 21 lines of code (we provide an excerpt in Table XI).

This part of our experimental study demonstrates how the expressiveness of Markov logic enables us to model data cleaning rules.

4.6 Summary

The results of our experimental study indicate that multiple types of data cleaning rules should be considered *holistically* (c.f., Section 4.2 and 4.3), which confirms previous research [Dallachiesa et al. 2013], [Fan and Geerts 2012]. Adding domain or structural knowledge about data into the Markov logic program improves the quality of data cleaning. Furthermore, joint inference enables us to achieve highly satisfactory results without having to define the order of the execution of the data cleaning rules. We find that joint modeling of data quality rules results in higher accuracy of data correction (c.f., Section 4.4). By using the probabilistic-logical framework of Markov logic, we benefit from its flexibility in constraint definition and joint inference over different data repair and matching rules. The direct translation of data cleaning rules into first-order logic (Markov logic formulas) simplifies the process of writing data cleaning routines (c.f., Section 4.5).

5. RELATED WORK

Our research builds on previous work from two areas: Data quality management and Statistical relational learning.

Data Quality Management: Holistic data cleaning methods based on integrity constraints and denial constraints with ad-hoc predicates have also been studied in [Chu et al. 2013]. This approach considers the generalization of

integrity constraints by translating them into denial constraints. However, denial constraints cannot express inclusion dependencies, hence in our work, we use the clausal form of the first-order predicate logic to express all kinds of integrity constraints. A generalization of dependencies was also proposed by the Llunatic system in [Geerts et al. 2013]. They introduce a new language based on equality generated dependencies to standardize the way in which to express intra- and inter-dependencies. The NADEEFF system from [Dallachiesa et al. 2013] is the system closest to ours with regard to the coverage of requirements for data cleaning systems. Analogous to our system, they treat data quality rules holistically. In contrast to NADEEFF, we use first-order logic (declarative approach) to define all the kinds of data quality rules and, therefore do not need black-boxes in the form of user-defined functions. We perform data cleaning as an inference process on Markov networks. Statistical inference for data cleaning has been used by Mayfield and his team in [Mayfield et al. 2010]. Their system ERACER has been designed to perform missing values imputation. In our work, we also use statistical inference to predict missing values, repair data, and detect duplicate entries. We model data quality rules based on FDs, CFDs, and MDs. Applying machine learning for entity deduplication has been demonstrated in [Guo et al. 2010]. The work in [Beskales et al. 2010] uses a probabilistic model for duplicate detection with uncertain outcomes. Throughout our work we use SRL for detection of all possible data quality issues. Recently suggested in [Prokoshyna et al. 2015] logical and statistical data cleaning applies metric functional dependencies as integrity constraint interface and proposes a strategy to choose a high-quality minimal repair. Assessing the effectiveness of data cleaning by introducing the statistical distortion metric is investigated in [Dasu and Loh 2012]. The most recent work in data curation is the Data Timer System [Stonebraker et al. 2013], which is an end-to-end system that performs massive data curation and data deduplication by combining two elements: machine learning and expert (human) feedback. In our work, we do not require human input to achieve high accuracy results, instead, we fully rely on the MAP-inference results to predict errors. Using machine learning and likelihood methods for cleaning noisy databases by predicting possible data updates has been introduced in SCARE system [Yakout et al. 2013]. In contrast to SCARE, our solution is capable of modeling and predicting not only missing values imputation and data consistency but also data deduplication. Recently proposed declarative approach in [Burdick et al. 2015] to data deduplication adopts *link-to-source* constraints and theoretically proofs a connection of entity linking to a probabilistic framework based on MLN. In our work, we empirically show that MLN based data cleaning is a natural fit for solving data quality issues.

Statistical Relational Learning: An advantage of probabilistic modeling for data quality has been investigated in [Naus et al. 1972] and [Chen et al. 2011]. Markov logic as a formalism for joint inference has been successfully used in a number of tasks, including natural language processing [Riedel and Meza-Ruiz 2008], ontology alignment, data integration [Noessner et al. 2013] and co-reference resolution [Poon and Domingos 2008]. These research results demonstrate the advantage of joint modeling vs. pipeline execution. We are the first to apply this formalism to data cleaning and to show the benefits of a joint data cleaning approach that uses MLNs as a framework for interacting data quality rules.

6. CONCLUSION

We presented a declarative data cleaning approach based on statistical relational learning and probabilistic inference. We demonstrated how integrity constraints, expressed as first-order logic formulas, are translated into probabilistic logical languages, allowing us to reason over inconsistencies or duplicates in a probabilistic way. Our approach allows the usage of probabilistic joint inference over interleaved data cleaning rules to improve data quality. By using a declarative probabilistic-logical formalism such as Markov logic, we are able to incorporate more semantic constraints and, therefore, extend traditional data quality rules. The results that we have presented in this paper indicate that taking a holistic view on data cleaning and that modeling this intuition within a Markov logic framework is a feasible and effective means to create data cleaning systems.

REFERENCES

Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. *Foundations of Databases*. Addison-Wesley.

Arvind Arasu, Christopher Ré, and Dan Suciu. 2009. Large-Scale Deduplication with Constraints Using Dedupalog *(ICDE '09)*. IEEE Computer Society, 12. DOI: http://dx.doi.org/10.1109/ICDE.2009.43

George Beskales, Mohamed A Soliman, Ihab F Ilyas, Shai Ben-David, and Yubin Kim. 2010. ProbClean: A probabilistic duplicate detection system. In *ICDE*. IEEE.

Philip Bohannon, Wenfei Fan, Michael Flaster, and Rajeev Rastogi. 2005. A cost-based model and effective heuristic for repairing constraints by value modification. In *ACM SIGMOD*. ACM.

Douglas Burdick, Ronald Fagin, Phokion G. Kolaitis, Lucian Popa, and Wang-Chiew Tan. 2015. A Declarative Framework for Linking Entities. In *18th International Conference on Database Theory (ICDT 2015)*, Marcelo Arenas and Martín Ugarte (Eds.), Vol. 31. DOI: http://dx.doi.org/10.4230/LIPIcs.ICDT.2015.25

Kuang Chen, Harr Chen, Neil Conway, Joseph M Hellerstein, and Tapan S Parikh. 2011. Usher: Improving data quality with dynamic forms. *Knowledge and Data Engineering, IEEE Transactions on* 23, 8 (2011), 1138–1153.

Wenguang Chen, Wenfei Fan, and Shuai Ma. 2009. Incorporating Cardinality Constraints and Synonym Rules into Conditional Functional Dependencies. *Inf. Process. Lett.* 109, 14 (June 2009), 783–789. DOI: http://dx.doi.org/10.1016/j.ipl.2009.03.021

Xu Chu, Ihab F Ilyas, and Paolo Papotti. 2013. Holistic data cleaning: Putting violations into context. In *Data Engineering (ICDE), 2013 IEEE*. 458–469.

US National Research Council. 2013. *Frontiers in Massive Data Analysis*. The National Academies Press. http://www.nap.edu/openbook.php?record_id=18374

Michele Dallachiesa, Amr Ebaid, Ahmed Eldawy, Ahmed Elmagarmid, Ihab F. Ilyas, Mourad Ouzzani, and Nan Tang. 2013. NADEEF: A Commodity Data Cleaning System *(SIGMOD '13)*. ACM, 541–552. DOI: http://dx.doi.org/10.1145/2463676.2465327

Tamraparni Dasu and Ji Meng Loh. 2012. Statistical distortion: Consequences of data cleaning. *Proceedings of the VLDB Endowment* 5, 11 (2012), 1674–1683.

Pedro Domingos and Daniel Lowd. 2009. Markov logic: An interface layer for artificial intelligence. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 3, 1 (2009), 1–155.

Wayne W. Eckerson. 2002. Data Warehousing Special Report: Data quality and the bottom line. (2002). http://download.101com.com/pub/tdwi/Files/DQReport.pdf Online; accessed 12-July-2014.

Ronald Fagin. 1982. Horn Clauses and Database Dependencies. *J. ACM* 29, 4 (Oct. 1982), 952–985. DOI: http://dx.doi.org/10.1145/322344.322347

Wenfei Fan and Floris Geerts. 2012. Foundations of Data Quality Management. *Synthesis Lectures on Data Management* 4, 5 (2012), 1–217.

Wenfei Fan, Shuai Ma, Nan Tang, and Wenyuan Yu. 2014. Interaction Between Record Matching and Data Repairing. *J. Data and Information Quality* 4, 4, Article 16 (May 2014), 38 pages. DOI: http://dx.doi.org/10.1145/2567657

Pedro F Felzenszwalb and Daniel P Huttenlocher. 2006. Efficient belief propagation for early vision. *International journal of computer vision* 70, 1 (2006), 41–54.

Floris Geerts, Mecca Giansalvatore, Paolo Papotti, and Donatello Santore. 2013. The Llunatic Data Cleaning Framework. *PVLDB* 6, 9 (2013), 625–636.

Michael R Genesereth and Nils J Nilsson. 1987. Logical Foundations of Artificial Intelligence. *Intelligence. Morgan Kaufmann* (1987).

Lise Getoor and Ben Taskar. 2007. *Introduction to statistical relational learning*. MIT press.

Songtao Guo, Xin Luna Dong, Divesh Srivastava, and Remi Zajac. 2010. Record linkage with uniqueness constraints and erroneous values. *Proceedings of the VLDB Endowment* 3, 1-2 (2010), 417–428.

James R. Lewis, Brian S. Utesch, and Deborah E. Maher. 2013. UMUX-LITE: When There's No Time for the SUS *(CHI '13)*. 4. DOI: http://dx.doi.org/10.1145/2470654.2481287

Jixue Liu, Jiuyong Li, Chengfei Liu, Yongfeng Chen, and others. 2012. Discover dependencies from data — a review. *IEEE Transactions on Knowledge and Data Engineering* 24, 2 (2012), 251–264.

Chris Mayfield, Jennifer Neville, and Sunil Prabhakar. 2010. ERACER: A Database Approach for Statistical Inference and Data Cleaning *(SIGMOD '10)*. 12. DOI: http://dx.doi.org/10.1145/1807167.1807178

Joseph I Naus, Thomas G Johnson, and Ramiro Montalvo. 1972. A Probabilistic Model for Identifying Errors in Data Editing. *J. Amer. Statist. Assoc.* 67, 340 (1972), 943–950. DOI: http://dx.doi.org/10.1080/01621459.1972.10481323

Jan Noessner, Mathias Niepert, and Heiner Stuckenschmidt. 2013. RockIt: Exploiting Parallelism and Symmetry for MAP Inference in Statistical Relational Models. In *AAAI*.

Hoifung Poon and Pedro Domingos. 2008. Joint unsupervised coreference resolution with Markov logic. In *EMNLP*.

Nataliya Prokoshyna, Jaroslaw Szlichta, Fei Chiang, Renée J Miller, and Divesh Srivastava. 2015. Combining quantitative and logical data cleaning. *Proceedings of the VLDB Endowment* 9, 4 (2015), 300–311.

Sebastian Riedel. 2008. Improving the accuracy and Efficiency of MAP Inference for Markov Logic. In *Proceedings of the UAI '08*. 468–475.

Sebastian Riedel and Ivan Meza-Ruiz. 2008. Collective Semantic Role Labelling with Markov Logic. In *CoNLL' 08*.

Alexander Schwing, Tamir Hazan, Marc Pollefeys, and Raquel Urtasun. 2011. Distributed message passing for large scale graphical models. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, and Kuansan Wang. 2015. An Overview of Microsoft Academic Service (MAS) and Applications. W3C.

David Alexander Sontag. 2010. *Approximate inference in graphical models using LP relaxations*. Ph.D. Dissertation. Massachusetts Institute of Technology.

Michael Stonebraker, George Beskales, Alexander Pagan, Daniel Bruckner, Mitch Cherniack, Shan Xu, Verisk Analytics, Ihab F. Ilyas, and Stan Zdonik. 2013. Data Curation at Scale: The Data Tamer System. In *In CIDR 2013*.

Mohamed Yakout, Laure Berti-Équille, and Ahmed K. Elmagarmid. 2013. Don'T Be SCAREd: Use SCalable Automatic REpairing with Maximal Likelihood and Bounded Changes. In *Proceedings of the ACM SIGMOD*. DOI:http://dx.doi.org/10.1145/2463676.2463706